DA-660-8/16-LX User's Manual

Fourth Edition, April 2008

www.moxa.com/product

© 2008 Moxa Inc., all rights reserved.
Reproduction without permission is prohibited.

DA-660-8/16-LX User's Manual

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

Copyright Notice

Copyright © 2008 Moxa Inc
All rights reserved.
Reproduction without permission is prohibited.

Trademarks

MOXA is a registered trademark of the Moxa Group.
All other trademarks or registered marks in this manual belong to their respective manufacturers.

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.

Moxa provides this document "as is," without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Technical Support Contact Information
www.moxa.com/support

Moxa Americas:
Toll-free: 1-888-669-2872
Tel: +1-714-528-6777
Fax: +1-714-528-6778

Moxa China (Shanghai office):
Toll-free: 800-820-5036
Tel: +86-21-5258-9955
Fax: +86-10-6872-3958

Moxa Europe:
Tel: +49-89-3 70 03 99-0
Fax: +49-89-3 70 03 99-99

Moxa Asia-Pacific:
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

Table of Contents

Chapter 1 Introduction 1-1

Overview 1-2
Software Architecture 1-2
- Journaling Flash File System (JFFS2) 1-3
- Software Package 1-4

Chapter 2 Getting Started 2-1

Powering on the DA-660 2-2
Connecting the DA-660 to a PC 2-2
- Serial Console 2-2
- Telnet Console 2-3
- SSH Console 2-4

Configuring the Ethernet Interface 2-5
- Modifying Network Settings with the Serial Console 2-5
- Modifying Network Settings over the Network 2-6

Test Program—Developing Hello.c 2-6
- Installing the Tool Chain (Linux) 2-7
- Checking the Flash Memory Space 2-7
- Compiling Hello.c 2-8
- Uploading and Running the “Hello” Program 2-8

Developing Your First Application 2-9
- Testing Environment 2-9
- Compiling tcps2.c 2-9
- Uploading and Running the “tcps2-release” Program 2-10
- Testing Procedure Summary 2-13

Chapter 3 Managing Embedded Linux 3-1

System Version Information 3-2
System Image Backup 3-2
- Upgrading the Firmware 3-2
- Loading Factory Defaults 3-4

Enabling and Disabling Daemons 3-4
Setting the Run-level 3-6
Adjusting the System Time 3-7
- Setting the Time Manually 3-7
- NTP Client 3-8
- Updating the Time Automatically 3-8

Cron—Daemon to Execute Scheduled Commands 3-9
Timezone Setting 3-9

Chapter 4 Managing Communications 4-1

Telnet / FTP 4-2
DNS 4-2
Web Service—Apache 4-3
IPTABLES 4-5
NAT 4-9
- NAT Example 4-9
- Enabling NAT at Bootup 4-10

Dial-up Service—PPP 4-10

PPPoE 4-13
NFS (Network File System) 4-15
Setting up the DA-660 as a NFS Server 4-15
Setting up the DA-660 as a NFS Client 4-16
Mail 4-17
SNMP 4-17
OpenVPN 4-18

Chapter 5 Programmer's Guide 5-1
Flash Memory Map 5-2
Linux Tool Chain Introduction 5-2
Debugging with GDB 5-3
Device API 5-4
RTC (Real Time Clock) 5-4
Buzzer 5-4
WDT (Watch Dog Timer) 5-5
UART 5-9
LCM 5-10
KeyPad 5-10
Makefile Example 5-11

Chapter 6 UC Finder 6-1
Windows UC Finder 6-2
Installing the Software 6-2
Broadcast Search 6-3
Linux UC Finder 6-4

Appendix A System Commands A-1
busybox (V0.60.4): Linux normal command utility collection A-1
File Manager A-1
Editor A-1
Network A-1
Process A-2
Other A-2
Moxa Special Utilities A-2

Appendix B Windows Tool Chain Introduction B-1
Introduction B-2
Installation Procedure B-3
Using the BASH Shell B-5
Compilation B-6
GDB Debug Tool—Insight B-8

Appendix C Using the LCM Keypad C-1

1

Introduction

Welcome to the DA-660 RISC-based Communication Platforms. Features include RS-232/422/485 serial ports, and dual 10/100 Mbps Ethernet ports, making the DA-660 ideal for your embedded applications.

The following topics are covered in this chapter:

- **Overview**
- **Software Architecture**
  - Journaling Flash File System (JFFS2)
  - Software Package

Overview

The DA-660 computer is ideal for embedded applications. This computer features a RISC CPU, RAM memory, and communication ports that connect to RS-232/422/485 serial devices and dual 10/100 Mbps Ethernet.

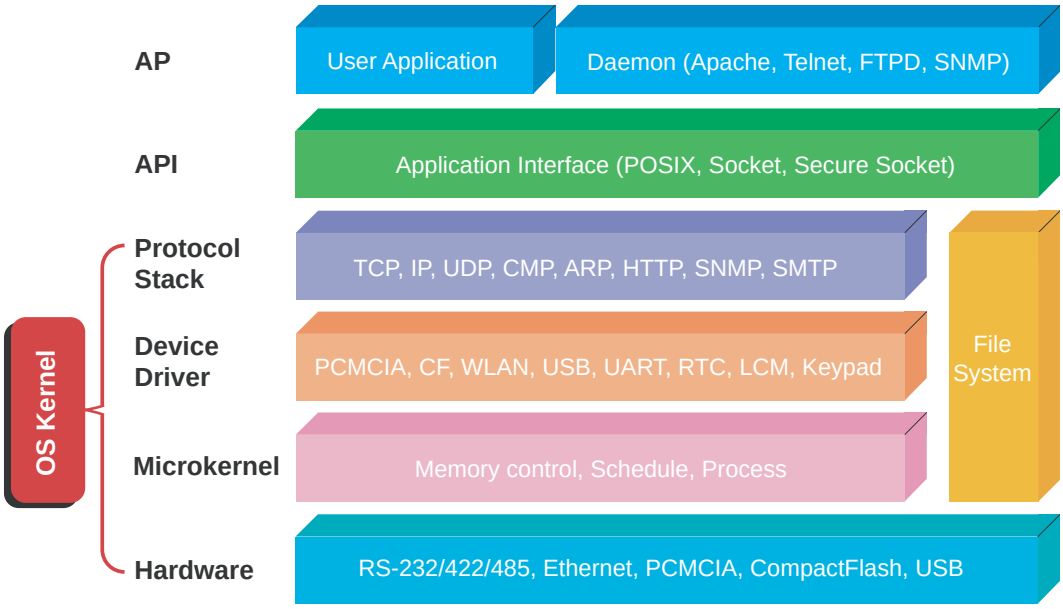
The Da-660 computer uses an Intel XScale IXP-422 266 Mhz RISC CPU. Unlike the X86 CPU, which uses a CISC design, the RISC architecture and modern semiconductor technology provide this computer with a powerful computing engine and communication functions, but without generating a lot of heat. Built-in 32 MB NOR Flash ROM and 128 MB SDRAM give you enough memory to install your application software directly on the computer. In addition, dual LAN ports are built right into the RISC CPU. The network capability, in combination with its control over serial devices, makes the Da-660 an ideal communication platform for data acquisition and industrial control applications.

DA-660's pre-installed Linux operating system (OS) provides an open software operating system for your software program development. Software written for desktop PCs can be easily ported to the computer with a GNU cross compiler, without needing to modify the source code. The OS, device drivers (e.g., Keypad, LCM, and Buzzer control) and your own applications, can all be stored in the NOR Flash memory.

The DA-660 Linux Series (referred to as DA-660, or the target computer) comes in two models: the DA-660-8-LX has 8 serial ports, and the DA-660-16-LX has 16 serial ports. Both models have exactly the same software and hardware features.

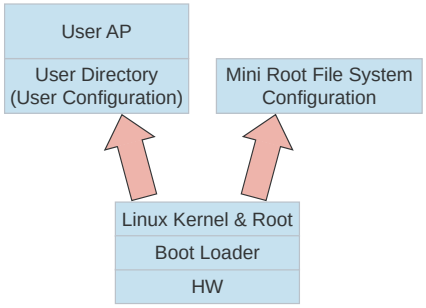
Software Architecture

The Linux operating system that is pre-installed in the DA-660 follows standard Linux architecture, making it easy to accept programs that follow the POSIX standard. Program porting is done with the GNU Tool Chain provided by Moxa. In addition to Standard POSIX APIs, device drivers for the LCM, Buzzer and Keypad controls, and UART are also included in the Linux OS.



The DA-660's built-in Flash ROM is partitioned into **Boot Loader, Linux Kernel, Mini Root File System,** and **User Root File System** partitions.

In order to prevent user applications from crashing the Root File System, the DA-660 uses a specially designed **Mini File System with Protected Configuration** for emergency use. This **Mini File System** comes with serial and Ethernet communication capability for users to load the **Factory Default Image** file. The Mini File System will only be activated if the boot loader fails to load the User Root File System.



To improve system reliability, the DA-660 has a built-in mechanism that prevents the system from crashing. The procedure is as follows.

When the Linux kernel boots up, the kernel will mount the root file system, and then enable services and daemons. During this time, the kernel will start searching for system configuration parameters via ***rc*** or ***inittab***.

Normally, the kernel uses the User Root File System to boot up the system. The Mini Root File System is protected, and cannot be changed by the user, and thus provides a safe zone. The kernel will only use the Mini Root File System when the User Root File System crashes.

For more information about the memory map and programming in DA-660, refer to Chapter 5, ***Programmer's Guide*.**

Journaling Flash File System (JFFS2)

The User Root File System in the flash memory is formatted with the **Journaling Flash File System (JFFS2)**. The formatting process places a compressed file system in the flash memory, transparent to the user.

The Journaling Flash File System (JFFS2), which was developed by Axis Communications in Sweden, places a file system directly on the flash, instead of emulating a block device. It is designed for use on flash-ROM chips and recognizes the special write requirements of a flash-ROM chip. JFFS2 implements wear-leveling to extend the life of the flash disk, and stores the flash directory structure in the RAM. A log-structured file system is maintained at all times. The system is always consistent, even if it encounters crashes or improper power-downs, and does not require ***fsck*** (file system check) on boot-up.

JFFS2 is the newest version of JFFS. It provides improved wear-leveling and garbage-collection performance, improved RAM footprint and response to system-memory pressure, improved concurrency and support for suspending flash erases, marking of bad sectors with continued use of the remaining good sectors that enhances the write-life of the devices, native data compression inside the file system design, and support for hard links.

The key features of JFFS2 are:

- Targets the Flash ROM directly
- Robustness
- Consistency across power failures
- No integrity scan (fsck) is required at boot time after normal or abnormal shutdown
- Explicit wear leveling
- Transparent compression

Although JFFS2 is a journaling file system, it does not exclude the loss of data. The file system will remain in a consistent state across power failures and will always be mountable. However, if the board is powered down during a write operation, then the incomplete write operation will be rolled back on the next boot, but write operations that have already been completed will not be

affected.

Additional information about JFFS2 is available at:
http://sources.redhat.com/jffs2/jffs2.pdf
http://developer.axis.com/software/jffs/
http://www.linux-mtd.infradead.org/

Software Package

Boot Loader	Redboot (V1.92)
Kernel	Monta Vista embedded Linux 2.4.18
Protocol Stack	ARP, PPP, CHAP, PAP, IPv4, ICMP, TCP, UDP, DHCP, FTP, SNMP V1/V3, HTTP, NTP, NFS, SMTP, SSH 1.0/2.0, SSL, Telnet, PPPoE, OpenVPN
File System	JFFS2, NFS, Ext2, Ext3, VFAT/FAT
OS shell command	bash
Busybox	Linux normal command utility collection
Utilities	
tinylogin	login and user manager utility
telnet	telnet client program
ftp	FTP client program
smtpclient	email utility
scp	Secure file transfer Client Program
Daemons	
pppd	dial in/out over serial port daemon
snmpd	snmpd agent daemon
telnetd	telnet server daemon
inetd	TCP server manager program
ftpd	ftp server daemon
apache	web server daemon
sshd	secure shell server
nfs-user-server	network file system server
openvpn	virtual private network
openssl	open SSL
Linux Tool Chain	
GCC (V3.3.2)	C/C++ PC Cross Compiler
GDB (V5.3)	Source Level Debug Server
Glibc (V2.2.5)	POSIX standard C library
Windows Tool Chain	
GCC (V3.3.2)	C/C++ PC Cross Compiler
GDB (V5.3)	Source Level Debug Server
Glibc(V2.2.5)	POSIX standard C library
Insight (V60.1)	Windows environment source level debug utility

2

Getting Started

In this chapter, we explain how to connect the DA-660, turn on the power, and then get started using the programming and other functions.

The following topics are covered in this chapter:

- **Powering on the DA-660**
- **Connecting the DA-660 to a PC**
  - Serial Console
  - Telnet Console
  - SSH Console
- **Configuring the Ethernet Interface**
  - Modifying Network Settings with the Serial Console
  - Modifying Network Settings over the Network
- **Test Program—Developing Hello.c**
  - Installing the Tool Chain (Linux)
  - Checking the Flash Memory Space
  - Compiling Hello.c
  - Uploading and Running the “Hello” Program
- **Developing Your First Application**
  - Testing Environment
  - Compiling tcps2.c
  - Uploading and Running the “tcps2-release” Program
  - Testing Procedure Summary

Powering on the DA-660

Connect the SG wire to the Shielded Contact located on the upper left corner of the DA-660, and then power on the computer by connecting it to the power adaptor. It takes about 30 to 60 seconds for the system to boot up. Once the system is ready, the Ready LED will light up, and the model name of the computer will appear on the LCM display.

NOTE After connecting the DA-660 to the power supply, it takes about 30 to 60 seconds for the operating system to boot up. The green Ready LED will not turn on until the operating system is ready.

Connecting the DA-660 to a PC

There are two ways to connect the DA-660 to a PC: through the serial Console port or by using Telnet over the network.

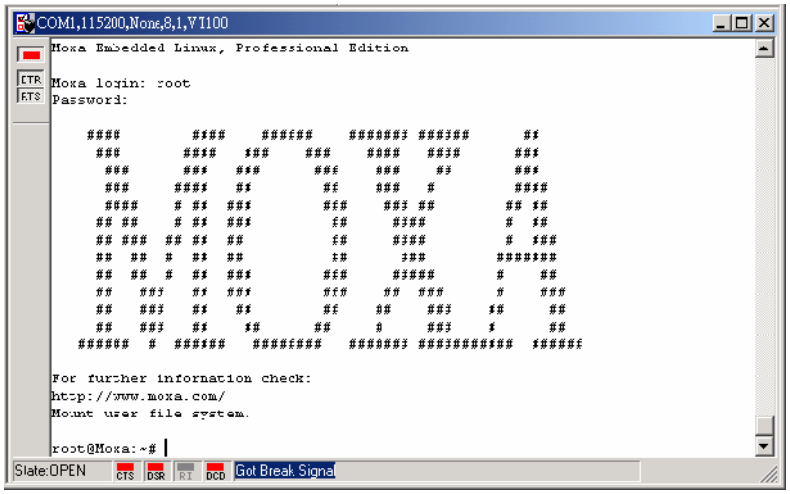
Serial Console

The serial console port gives users a convenient way of connecting to the DA-660's console utility. This method is particularly useful when using the computer for the first time. The signal is transmitted over a direct serial connection, and therefore there is no need to know DA-660's two IP addresses in order to connect to the serial console utility.

Use the serial console port settings shown below.

Baudrate	115200 bps
Parity	None
Data bits	8
Stop bits:	1
Flow Control	None
Terminal	VT100

Once the connection is established, the following window will open.



To log in, type the Login name and password as requested. The default values are both **root**:

```
Login:    root
Password: root
```

Telnet Console

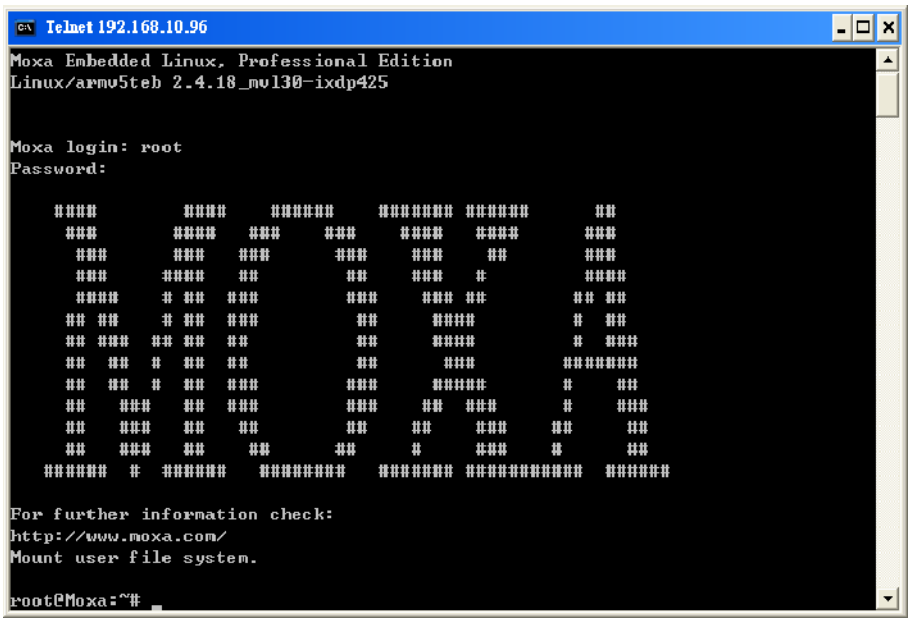
If you know at least one of the two IP addresses and netmasks, then you can use Telnet to connect to the DA-660's console utility. The default IP address and netmask for each of the two ports are given below:

	Default IP Address	Netmask
LAN 1	192.168.3.127	255.255.255.0
LAN 2	192.168.4.127	255.255.255.0

Use a cross-over Ethernet cable to connect directly from your PC to the DA-660. You should first modify your PC's IP address and netmask so that your PC is on the same subnet as one of the DA-660's two LAN ports. For example, if you connect to LAN 1, you can set your PC's IP address to 192.168.3.126 and netmask to 255.255.255.0. If you connect to the LAN 2, you can set your PC's IP address to 192.168.4.126 and netmask to 255.255.255.0.

To connect to a hub or switch connected to your local LAN, use a straight-through Ethernet cable. The default IP addresses and netmasks are shown above. To log in, type the Login name and password as requested. The default values are both **root**:

```
Login:    root
Password: root
```



You can proceed with configuring network settings of the target computer from the built-in **bash** shell. Configuration instructions are given in the next section.

ATTENTION

Serial Console Reminder

Remember to choose VT100 as the terminal type. Use the cable provided, CBL-RJ45F9-150, to connect to the serial console port.

Telnet Reminder

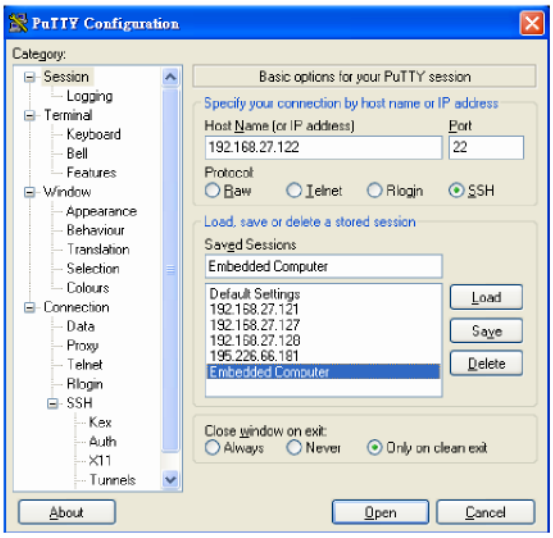
When connecting to the DA-660 over a LAN, you must configure your PC's Ethernet IP address to be on the same subnet as the DA-660 you wish to contact. If you do not get connected on the first try, re-check the serial and IP settings, and then unplug and re-plug the DA-660's power cord.

SSH Console

The DA-660 supports an SSH Console to offer users with better security options.

Windows Users

Click the link http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html to download PuTTY (free software) to set up an SSH console for the DA-660 in a Windows environment. The following figure shows a simple example of the configuration that is required.



Linux Users

From a Linux machine, use the ***ssh*** command to access the DA-660's Console utility via SSH.

```
#ssh 192.168.3.127
```

Select **yes** to complete the connection.

```
[root@bee_notebook root]# ssh 192.168.3.127
The authenticity of host '192.168.3.127 (192.168.3.127)' can't be established.
RSA key fingerprint is 8b:ee:ff:84:41:25:fc:cd:2a:f2:92:8f:cb:1f:6b:2f.
Are you sure you want to continue connection (yes/no)? yes_
```

NOTE SSH provides better security compared to Telnet for accessing the DA-660's Console utility over the network.

Configuring the Ethernet Interface

The network settings of the DA-660 can be modified with the serial Console, or online over the network.

Modifying Network Settings with the Serial Console

In this section, we use the serial console to configure the network settings of the target computer.

1. Follow the instructions given in a previous section to access the Console Utility of the target computer via the serial Console port, and then type **#cd /etc/network** to change directories.

```
root@Moxa:# cd /etc/network/
root@Moxa:/etc/network/#
```

2. Type **#vi interfaces** to edit the network configuration file with vi editor. You can configure the Ethernet ports of the DA-660 for **static** or **dynamic** (DHCP) IP addresses.

Static IP addresses:

As shown below, 4 network settings need to be modified: **address, network, netmask,** and **broadcast**. The default IP addresses are 192.168.3.127 for LAN1 and 192.168.4.127 for LAN2, with a default netmask of 255.255.255.0.

```
# We always want the loopback interface.

auto eth0 eth1 lo
iface lo inet loopback

# embedded ethernet LAN1
iface eth0 inet static
       address 192.168.3.127
       network 192.168.3.0
       netmask 255.255.255.0
       broadcast 192.168.3.255

# embedded ethernet LAN2
iface eth1 inet static
       address 192.168.4.127
       network 192.168.4.0
       netmask 255.255.255.0
       broadcast 192.168.4.255

# 802.11g Gigabyte Cardbus wireless card
#iface eth2 inet static
       address 192.168.5.127
       network 192.168.5.0
"/etc/network/interfaces" line 1 of 162 --0%--
```

Dynamic IP addresses:

By default, the DA-660 is configured for static IP addresses. To configure one or both LAN ports to request an IP address dynamically, replace **static** with **dhcp** and then delete the address, network, netmask, and broadcast lines.

Default Setting for LAN1	Dynamic Setting using DHCP
iface eth0 inet **static** address 192.168.3.127 network: 192.168.3.0 netmask 255.255.255.0 broadcast 192.168.3.255	iface eth0 inet **dhcp**

```
Auto eth0 eth1 lo
iface lo inet loopback

iface eth0 inet dhcp

iface eth1 inet dhcp
```

3. After modifying the boot settings of the LAN interface, issue the following command to activate the LAN settings immediately:

   #/etc/init.d/networking restart

NOTE After changing the IP settings, use the **networking restart** command to activate the new IP address. However, the LCM display will still show the old IP address. To update the LCM display, you will need to reboot the DA-660.

Modifying Network Settings over the Network

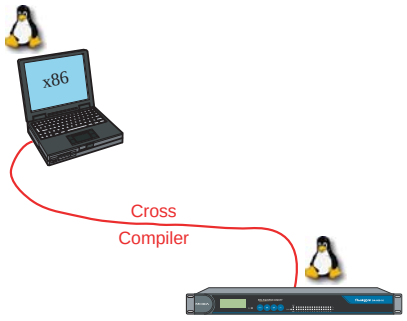
IP settings can be activated over the network, but the new settings will not be saved to the flash ROM without modifying the file **/etc/network/interfaces**.

For example, type the command **#ifconfig eth0 192.168.1.1** to change the IP address of LAN1 to 192.168.1.1.

```
root@Moxa:~# ifconfig eth0 192.168.1.1
root@Moxa:~#
```

Test Program—Developing Hello.c

In this section, we use the standard **Hello** programming example to illustrate developing a program for the DA-660. In general, program development for DA-660 involves the following seven steps.



Step 1:
Connect the DA-660 to a Linux PC.
Step 2:
Install Tool Chain (GNU Cross Compiler & glibc).
Step 3:
Set the cross compiler and glibc environment variables.
Step 4:
Code and compile the program.
Step 5:
Download the program to the DA-660 via FTP or NFS.
Step 6:
Debug the program
→ If bugs are found, return to Step 4.
→ If no bugs are found, continue with Step 7.
Step 7:
Back up the user directory (distribute the program to additional DA-660 units if needed).

Installing the Tool Chain (Linux)

The PC must have the Linux Operating System pre-installed before installing the DA-660 GNU Tool Chain. Redhat 7.3/8.0, Fedora Core, and compatible versions are recommended. The Tool Chain requires about 100 MB of hard disk space on your PC. The DA-660 Tool Chain software is located on the DA-660 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#rpm -ivh /mnt/cdrom/mxscaleb-3.3.2-6.i386.rpm
```

The Tool Chain will be installed automatically on your Linux PC within a few minutes. Before compiling the program, be sure to set the following path first, since the Tool Chain files, including the compiler, link, library, and include files are located in this directory.

```
PATH=/usr/local/mxscaleb/bin:$PATH
```

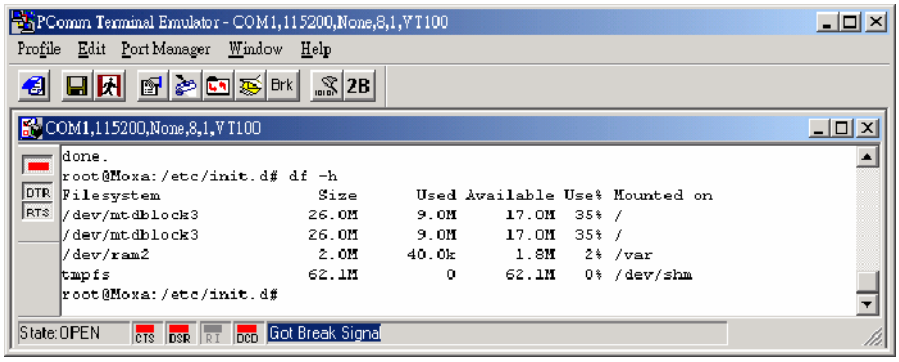
Setting the path allows you to run the compiler from any directory.

NOTE Refer to Appendix B for an introduction to the Windows Tool Chain. In this chapter, we use the Linux tool chain to illustrate the cross compiling process.

Checking the Flash Memory Space

If the flash memory is full, you will not be able to save data to the Flash ROM. Use the following command to calculate the amount of **Available** flash memory:

```
/>df -h
```



If there is not enough **Available** space for your application, you will need to delete some existing files. To do this, connect your PC to the DA-660 with the console cable, and then use the console utility to delete the files from the DA-660's flash memory.

NOTE If the flash memory is full, you will need to free up some memory space before saving files to the Flash ROM.

Compiling Hello.c

The package CD contains several example programs. Here we use **Hello.c** as an example to show you how to compile and run your applications. Type the following commands from your PC to copy the files used for this example from the CD to your computer's hard drive:

```
# cd /tmp/
# mkdir example
# cp –r /mnt/cdrom/example/* /tmp/example
```

To compile the program, go to the **Hello** subdirectory and issue the following commands:

```
#cd example/hello #make
```

You should receive the following response:

```
[root@localhost hello]# make
/usr/local/mxscaleb/bin/mxscaleb-gcc –o hello-release hello.c
/usr/local/mxscaleb/bin/mxscaleb-strip –s hello-release
/usr/local/mxscaleb/bin/mxscaleb-gcc –ggdb -o hello-debug hello.c
[root@localhost hello]# _
```

Next, execute the **hello.exe** file to generate **hello-release** and **hello-debug**, which are described below:

hello-release—an IXP platform execution file (created specifically to run on the DA-660)

hello-debug—an IXP platform GDB debug server execution file (see Chapter 5 for details about the GDB debug tool).

NOTE Be sure to type the **#make** command from within the **/tmp/example/hello** directory, since UC's tool chain places a specially designed **Makefile** in that directory. This special Makefile uses the mxscale-gcc compiler to compile the hello.c source code for the Xscale environment. If you type the **#make** command from any other directory, Linux will use the x86 compiler (for example, cc or gcc).

Refer to Chapter 5 to see a Makefile example.

Uploading and Running the "Hello" Program

Use the following commands to upload **hello-release** to the DA-660 via FTP.

1. From the PC, type:

   ```
   #ftp 192.168.3.127
   ```

2. Use the **bin** command to set the transfer mode to Binary mode, and the **put** command to initiate the file transfer:

   ```
   ftp> bin
   ftp> put hello-release
   ```

3. From the DA-660, type:

   ```
   # chmod +x hello-release
   # ./hello-release
   ```

The word **Hello** will be printed on the screen.

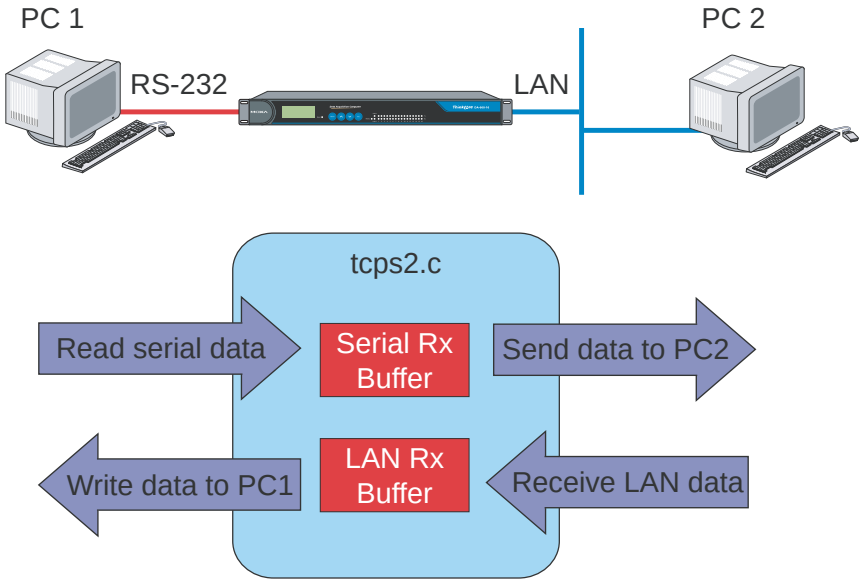
```
root@Moxa:~# ./hello-release
Hello
```

Developing Your First Application

We use the tcps2 example to illustrate how to build an application. The procedure outlined in the following subsections will show you how to build a TCP server program plus serial port communication that runs on the DA-660.

Testing Environment

The tcps2 example demonstrates a simple application program that delivers transparent, bi-directional data transmission between the DA-660's serial and Ethernet ports. As illustrated in the following figure, the purpose of this application is to transfer data between PC 1 and the DA-660 via an RS-232 connection. At the remote site, data can be transferred between the DA-660's Ethernet port and PC 2 over an Ethernet connection.



Compiling tcps2.c

The source code for the tcps2 example is located on the CD-ROM at **CD-ROM://example/TCPServer2/tcps2.c.** Use the following commands to copy the file to a specific directory on your PC. We use the directory **/home/da660/1st_application/**. Note that you need to copy 3 files—**Makefile, tcps2.c, tcpsp.c**—from the CD-ROM to the target directory.

```
#mount –t iso9660 /dev/cdrom /mnt/cdrom
#cp /mnt/cdrom/example/TCPServer2/tcps2.c/home/da660/1st_application/tcps2.c
#cp /mnt/cdrom/example/TCPServer2/tcpsp.c/home/da660/1st_application/tcpsp.c
#cp /mnt/cdrom/example/TCPServer2/Makefile.c/home/da660/1st_application/Makefile.c
```

Type **#make** to compile the example code:

You will get the following response, indicating that the example program was compiled successfully.

```
root@server11:/home/da660/1st_application
[root@server11 1st_application]# pwd
/home/da660/1st_application
[root@server11 1st_application]# 11
total 20
-rw-r--r--  1 root root  514 Nov 27 11:52 Makefile
-rw-r--r--  1 root root 4554 Nov 27 11:52 tcps2.c
-rw-r--r--  1 root root 6164 Nov 27 11:55 tcps2.c
[root@server11 1st_application]# make_
/usr/local/mxscaleb/bin/mxscaleb-gcc -o tcps2-release tcps2.c
/usr/local/mxscaleb/bin/mxscaleb-strip –s tcps2-release
/usr/local/mxscaleb/bin/mxscaleb-gcc -o tcpsp-release tcpsp.c
/usr/local/mxscaleb/bin/mxscaleb-strip –s tcpsp-release
/usr/local/mxscaleb/bin/mxscaleb-gcc –ggdb -o tcps2-debug tcps2.c
/usr/local/mxscaleb/bin/mxscaleb-gcc –ggdb -o tcpsp-debug tcpsp.c
You have new mail in /var/spool/mail/root
[root@server11 1st_application]# 1s
[root@server11 1st_application]# 11
total 92
-rw-r--r--  1 root root   514  Nov 27 11:52 Makefile
-rwxr-xr-x  1 root root 25843 Nov 27 12:03 tcps2-debug
-rwxr-xr-x  1 root root  4996  Nov 27 12:03 tcps2-release
-rw-r--r--  1 root root  4554  Nov 27 11:52 tcps2.c
-rwxr-xr-x  1 root root 26823 Nov 27 12:03 tcpsp-debug
-rwxr-xr-x  1 root root  5396  Nov 27 12:03 tcpsp-release
-rw-r--r--  1 root root  6164  Nov 27 11:55 tcpsp.c
[root@server11 1st_application]#
```

Two executable files, **tcps2-release** and **tcps2-debug**, are created.

tcps2-release—an IXP platform execution file (created specifically to run on the DA-660).

tcps2-debug—an IXP platform GDB debug server execution file (see Chapter 5 for details about the GDB debug tool).

NOTE If you get an error message at this point, it could be because you neglected to put tcps2.c and tcpsp.c in the same directory. The example Makefile we provide is set up to compile both tcps2 and tcpsp into the same project Makefile. Alternatively, you could modify the Makefile to suit your particular requirements.

Uploading and Running the "tcps2-release" Program

Use the following commands to use FTP to upload **tcps2-release** to the DA-660.

1. From the PC, type:

   ```
   #ftp 192.168.3.127
   ```

2. Next, use the **bin** command to set the transfer mode to **Binary,** and the **put** command to initiate the file transfer:

   ```
   ftp> bin
   ftp> put tcps2-release
   ```

```
root@server11:/home/da660/1st_application
 [root@server11 1st_application]# ftp 192.168.3.127
Connected to 192.168.3.127 220
Moxa FTP server (Version wu-2.6.1(2) Mon Nov 24 12:17:04 CST 2003) ready.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.3.127:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bin
200 Type set to I.
ftp> put tcps2-release
local: tcps2-release remote: tcps2-release
277 Entering Passive Mode (192.168.3.127.82.253)
150 Opening BINARY mode data connection for tcps2-release.
226 Transfer complete
4996 bytes sent in 0.00013 seconds (3.9e+04 Kbytes/s)
ftp> ls
227 Entering Passive Mode (192.168.3.127.106.196)
150 Opening ASCII mode data connection for /bin/ls.
-rw-------    1 root     root          899 Jun 10 08:11  bash_history
-rw-r--r--    1 root     root         4996 Jun 12 02:15 tcps2-release
226 Transfer complete
ftp>
```

3. From the DA-660, type:

```
# chmod +x tcps2-release
# ./tcps2-release &
```

```
192.168.3.127 - PuTTY
root@Moxa:~# ls –al
drwxr–xr-x  2 root  root     0 Jun 12 02:14
drwxr–xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rw-r--r--  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# chmod +x tcps2-release
root@Moxa:~# ls –al
drwxr–xr-x  2 root  root     0 Jun 12 02:14
drwxr–xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rwxr-xr-x  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~#
```

4. The program should start running in the background. Use either the **#jobs** or **#ps –ef** command to check if the tcps2 program is actually running in the background.

```
#jobs // use this command to check if the program is running
```

```
192.168.3.127 - PuTTY
root@Moxa:~# ls –al
drwxr–xr-x  2 root  root     0 Jun 12 02:14
drwxr–xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rw-r--r--  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# chmod +x tcps2-release
root@Moxa:~# ls –al
drwxr–xr-x  2 root  root     0 Jun 12 02:14
drwxr–xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rwxr-xr-x  1 root  root  4996 Jun 12 02:15 tcps2-release
```

```
root@Moxa:~# ./tcps2-release &
[1] 187
start
root@Moxa:~# jobs
[1]+  Running    ./tcps2-release &
root@Moxa:~#
```

NOTE Use the **kill** command for job number 1 to terminate this program: **#kill %1**

#ps -ef // use this command to check if the program is running

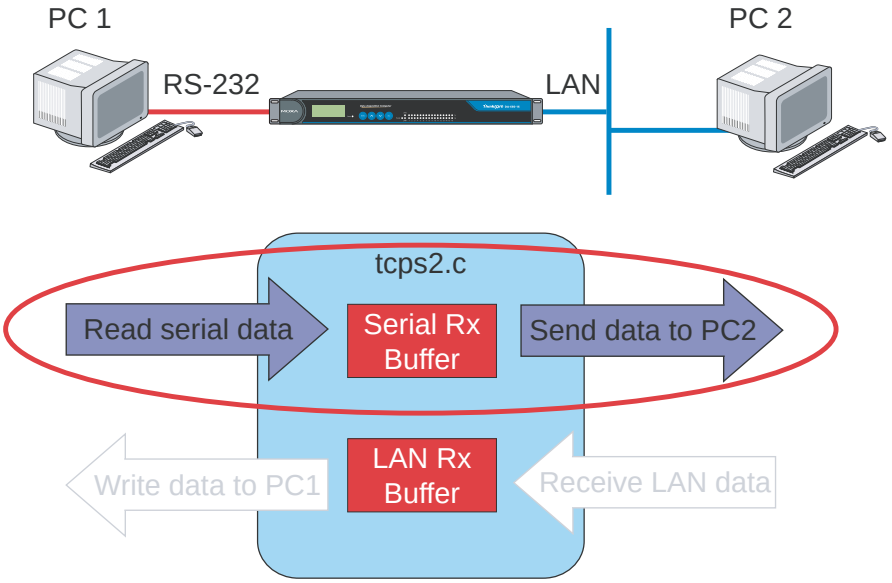
```
  192.168.3.127 - PuTTY
[1]+  Running    ./tcps2-release &
root@Moxa:~# ps –ef
PID  Uid    VmSize Stat Command
  1 root       1296 S    init
  2 root            S    [keventd]
  3 root            S    [ksoftirqd_CPU0]
  4 root            S    [kswapd]
  5 root            S    [bdflush]
  6 root            S    [kupdated]
  7 root            S    [mtdblockd]
  8 root            S    [khubd]
 10 root            S    [jffs2_gcd_mtd3]
 32 root            D    [ixp425_csr]
 34 root            S    [ixp425 eth0]
 36 root            D    [ixp425 eth1]
 38 root       1256 S    stdef
 46 root       1368 S    /usr/sbin/inetd
 52 root       4464 S    /usr/sbin/httpd
 53 nobody     4480 S    /usr/sbin/httpd
 54 nobody     4480 S    /usr/sbin/httpd
 64 nobody     4480 S    /usr/sbin/httpd
 65 nobody     4480 S    /usr/sbin/httpd
 66 nobody     4480 S    /usr/sbin/httpd
 88 bin        1460 S    /sbin/portmap
100 root       1556 S    /usr/sbin/rpc.statd
104 root       4044 S    /usr/sbin/snmpd –s –l /dev/null
106 root       2832 S    /usr/sbin/snmptrapd -s
135 root       1364 S    /sbin/cardmgr
139 root       1756 S    /usr/sbin/rpc.nfsd
141 root       1780 S    /usr/sbin/rpc.mountd
148 root       2960 S    /usr/sbin/sshd
156 root       1272 S    /bin/reportip
157 root       1532 S    /sbin/getty 115200 ttyS0
158 root       1532 S    /sbin/getty 115200 ttyS1
162 root       3652 S    /usr/sbin/sshd
163 root       2208 S    -bash
169 root       2192 S    ftpd: 192.168.3.110: root: IDLE
187 root       1264 S    ./tcps2-release
188 root       1592 S    ps -ef
root@Moxa:~#
```

NOTE Use the **kill -9** command for PID 187 to terminate this program: **#kill -9 187**

Testing Procedure Summary

1. Compile **tcps2.c** (`#make`).
2. Upload and run **tcps2-release** in the background (`#./tcps2-release &`).
3. Check that the process is running (`#jobs or #ps -ef`).
4. Use a serial cable to connect PC1 to the DA-660's first serial port.
5. Use an Ethernet cable to connect PC2 to the DA-660.
6. On PC1: If running Windows, use HyperTerminal (**38400, n, 8, 1**) to open COMn.
7. On PC2: Type `#telnet 192.168.3.127 4001`.
8. On PC1: Type some text and then press **Enter**.
9. On PC2: The text you typed on PC1 will appear on PC2's screen.

The testing environment is illustrated in the following figure. However, note that there are limitations to the example program **tcps2.c**.



NOTE

The **tcps2.c** application is a simple example designed to give users a basic understanding of the concepts involved in combining Ethernet communication and serial port communication. However, the example program has some limitations that make it unsuitable for real-life applications.

1. The serial port is in canonical mode and block mode, making it impossible to send data from the Ethernet side to the serial side (i.e., from PC 2 to PC 1 in our example).
2. The Ethernet side will not accept multiple connections.

3

Managing Embedded Linux

This chapter includes information about version control, deployment, updates, and peripherals. The information in this chapter is particularly useful when you need to run the same application on several DA-660 units.

The following topics are covered in this chapter:

- **System Version Information**
- **System Image Backup**
  - Upgrading the Firmware
  - Loading Factory Defaults
- **Enabling and Disabling Daemons**
- **Setting the Run-level**
- **Adjusting the System Time**
  - Setting the Time Manually
  - NTP Client
  - Updating the Time Automatically
- **Cron—Daemon to Execute Scheduled Commands**
- **Timezone Setting**

System Version Information

To determine the hardware capability of your DA-660, and what kind of software functions are supported, check the version numbers of your DA-660's hardware, kernel, and user file system. Contact Moxa to determine the hardware version. You will need the **Production S/N** (Serial number), which is located on the DA-660's bottom label.

To check the kernel version, type:
#kversion

To check the user file system version, type:
#fsversion

```
192.168.3.127 - PuTTY
root@Moxa:~# kversion
1.0
root@Moxa:~# fsversion
1.0
```

NOTE The kernel version and user file system version numbers are the same for the factory default configuration. Even if you download the latest firmware version from Moxa's website and then upgrade the DA-660's firmware, the two version numbers will still be the same.

However, to help users define the user file system, the kernel and user file system are separate, and hence could have different version numbers. For this reason, we provide two utilities, called **kversion** and **fsversion** that allow you to check the version numbers of the kernel and file system, respectively.

System Image Backup

Upgrading the Firmware

The DA-660's BIOS, kernel, mini file system, and user file system are combined into one firmware file, which can be downloaded from Moxa's website (www.moxa.com). The name of the file has the form **da660-x.x.x.frm**, with "x.x.x" indicating the firmware version. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the DA-660 unit via a serial Console or Telnet Console connection.

ATTENTION

Upgrading the firmware will erase all data on the Flash ROM

If you are using the ramdisk to store code for your applications, beware that updating the firmware will erase all of the data on the Flash ROM. You should back up your application files and data before updating the firmware.

Since different Flash disks have different sizes, it's a good idea to check the size of your Flash disk before upgrading the firmware, or before using the disk to store your application and data files. Use the #df –h command to list the size of each memory block, and the free space available in each block.

```
192.168.3.127 - PuTTY
root@Moxa:~# df –h
Filesystem          Size      Used Available Use% Mounted on
/dev/ mtdblock3     26.0M     8.9M     17.1M 34%  /
/dev/mtdblock3      26.0M     8.9M     17.1M 34%  /
/dev/ram2           2.0M     40.0k      1.8M   2%  /var
tmpfs               62.1M        0     62.1M   0%  /dev/shm
root@Moxa:~# upramdisk
root@Moxa:~# df –h
Filesystem          Size      Used Available Use% Mounted on
/dev/ mtdblock3     26.0M     8.9M     17.1M 34%  /
/dev/mtdblock3      26.0M     8.9M     17.1M 34%  /
/dev/ram2           2.0M     40.0k      1.8M   2%  /var
tmpfs               62.1M        0     62.1M   0%  /dev/shm
/dev/ram1           29.0M     13.0k     27.5M   0%  /mnt/ramdisk
root@Moxa:~# cd /mnt/ramdisk
root@Moxa:/mnt/ramdisk#
```

The following instructions give the steps required to save the firmware file to the DA-660's RAM disk, and then upgrade the firmware.

1. Type the following commands to enable the RAM disk:

   #upramdisk
   #cd /mnt/ramdisk

2. Type the following commands to use the DA-660's built-in FTP client to transfer the firmware file (**da660-x.x.x.frm**) from the PC to the DA-660:

   /mnt/ramdisk> ftp <destination PC's IP> Login Name: xxxx
   Login Password: xxxx
   ftp> bin
   ftp> get da660-x.x.x.frm

```
192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# ftp 192.168.3.193
Connected to 192.168.3.193 (192.168.3.193).
220 TYPSoft FTP Server 1.10 ready…
Name (192.168.3.193:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd newsw
250 CWD command successful. "/C:/ftproot/newsw/" is current directory.
ftp> bin
200 Type set to I.
ftp> ls
200 Port command successful.
150 Opening data connection for directory list.
drw-rw-rw-    1 ftp  ftp     0 Nov 30 10:03 .
drw-rw-rw-    1 ftp  ftp     0 Nov 30 10:03 .
-rw-rw-rw-    1 ftp  ftp  13167772 Nov 29 10:24 DA660-1.0.frm
-rw-rw-rw-    1 ftp  ftp   8778996 Nov 29 10:24 DA660_rootdisk-1.0.frm
226 Transfer complete.
ftp> get DA660-1.0.frm
local: DA660-1.0.frm remote: DA660-1.0.frm
200 Port command successful.
150 Opening data connection for DA660-1.0.frm
226 Transfer complete.
13167772 bytes received in 2.17 secs (5925.8 kB/s)
ftp>
```

3. Next, use the **upfirm** command to upgrade the kernel and root file system:

   #upfirm da660-x.x.x.frm

```
  192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# upfirm DA660-1.0.frm
Upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file conext is OK.
This step destroy all your firmware.
Do you want to continue it ? (Y/N) : Y
Now upgrade the file [redboot].
Format MTD device [/dev/mtd0] ...
MTD device [/dev/mtd0] erase 128 Kibyte @ 60000 - 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] ...
MTD device [/dev/mtd1] erase 128 Kibyte @ 100000 - 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [mini-file-system].
Format MTD device [/dev/mtd2] ...
MTD device [/dev/mtd2] erase 128 Kibyte @ 400000 - 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [user-file-system].
Format MTD device [/dev/mtd3] ...
MTD device [/dev/mtd3] erase 128 Kibyte @ 1a0000 - 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [directory].
Format MTD device [/dev/mtd6] ...
MTD device [/dev/mtd6] erase 128 Kibyte @ 20000 - 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the new configuration file.
Upgrade the firmware is OK.
Please press any key to reboot system.
```

Loading Factory Defaults

The easiest way to load factory defaults is to update the firmware (follow the instructions in the previous section to upgrade the firmware).

Note that if your user file is not working properly, the system will mount the Mini File System. In this case, you will need to load factory defaults to resume normal operation.

Enabling and Disabling Daemons

The following daemons are enabled when the DA-660 boots up for the first time.

snmpd .....SNMP Agent daemon
telnetd .....Telnet Server / Client daemon
inetd .....Internet Daemons
ftpd.....FTP Server / Client daemon
sshd .....Secure Shell Server daemon
httpd.....Apache WWW Server daemon
nfsd.....Network File System Server daemon

Type the command **ps –ef** to list all processes that are currently running.

```
192.168.3.127 - PuTTY
root@Moxa:~# cd /etc
root@Moxa:/etc# ps –ef
  PID  Uid    VmSize Stat Command
    1 root       1296 S    init
    2 root            S    [keventd]
    3 root            S    [ksoftirqd_CPU0]
    4 root            S    [kswapd]
    5 root            S    [bdflush]
    6 root            S    [kupdated]
    7 root            S    [mtdblockd]
    8 root            S    [khubd]
   10 root            S    [jffs2_gcd_mtd3]
   32 root            D    [ixp425_csr]
   34 root            S    [ixp425 eth0]
   38 root       1256 S    stdef
   36 root            S    [ixp425 eth1]
   47 root       1368 S    /usr/sbin/inetd
   53 root       4464 S    /usr/sbin/httpd
   54 nobody     4480 S    /usr/sbin/httpd
   64 nobody     4480 S    /usr/sbin/httpd
   65 nobody     4480 S    /usr/sbin/httpd
   66 nobody     4480 S    /usr/sbin/httpd
   67 nobody     4480 S    /usr/sbin/httpd
   92 bin        1460 S    /sbin/portmap
  104 root       1556 S    /usr/sbin/rpc.statd
  108 root       4044 S    /usr/sbin/snmpd –s –l /dev/null
  110 root       2828 S    /usr/sbin/snmptrapd –s
  139 root       1364 S    /sbin/cardmgr
  143 root       1756 S    /usr/sbin/rpc.nfsd
  145 root       1780 S    /usr/sbin/rpc.mountd
  152 root       2960 S    /usr/sbin/sshd
  160 root       1272 S    /bin/reportip
  161 root       3464 S    /bin/massupfirm
  162 root       1532 S    /sbin/getty 115200 ttyS01
  163 root       1532 S    /sbin/getty 115200 ttyS1
  166 root       3464 S    /bin/massupfirm
  167 root       3464 S    /bin/massupfirm
  170 root       3652 S    /usr/sbin/sshd
  171 root       2196 S    –bash
  182 root       1592 S    ps –ef
root@Moxa:/ect#
```

To run a private daemon, you can edit the file rc.local, as follows:

```
#cd /etc/rc.d
#vi rc.local
```

```
192.168.3.127 - PuTTY
root@Moxa:~# cd /etc/rc.d
root@Moxa:/etc/rc.d# vi rc.local
```

Next, add the application daemon that you want to run, to this file. We use the example program **tcps2-release**, and configure it to run in the background.

```
192.168.3.127 - PuTTY
# !/bin/sh
# Add you want to run daemon
/root/tcps2-release &~
```

View the list of enabled daemons, after reboot, to see whether your application daemon is enabled.

```
192.168.3.127 - PuTTY
root@Moxa:~# ps –ef
  PID  Uid    VmSize Stat Command
    1 root      1296 S    init
    2 root           S    [keventd]
    3 root           S    [ksoftirqd_CPU0]
    4 root           S    [kswapd]
    5 root           S    [bdflush]
    6 root           S    [kupdated]
    7 root           S    [mtdblockd]
    8 root           S    [khubd]
   10 root           S    [jffs2_gcd_mtd3]
   32 root           D    [ixp425_csr]
   34 root           S    [ixp425 eth0]
   36 root           S    [ixp425 eth1]
   38 root      1256 S    stdef
   47 root      1368 S    /usr/sbin/inetd
   53 root      4464 S    /usr/sbin/httpd
   63 nobody    4480 S    /usr/sbin/httpd
   64 nobody    4480 S    /usr/sbin/httpd
   65 nobody    4480 S    /usr/sbin/httpd
   66 nobody    4480 S    /usr/sbin/httpd
   67 nobody    4480 S    /usr/sbin/httpd
   92 bin       1460 S    /sbin/portmap
   97 root      1264 S    /root/tcps2-release
  105 root      1556 S    /usr/sbin/rpc.statd
  109 root      4044 S    /usr/sbin/snmpd –s –l /dev/null
  111 root      2832 S    /usr/sbin/snmptrapd –s
  140 root      1364 S    /sbin/cardmgr
  144 root      1756 S    /usr/sbin/rpc.nfsd
  146 root      1780 S    /usr/sbin/rpc.mountd
  153 root      2960 S    /usr/sbin/sshd
  161 root      1272 S    /bin/reportip
  162 root      3464 S    /bin/massupfirm
  163 root      1532 S    /sbin/getty 115200 ttyS0
  164 root      1532 S    /sbin/getty 115200 ttyS1
  166 root      3464 S    /bin/massupfirm
  168 root      3464 S    /bin/massupfirm
  171 root      3652 S    /usr/sbin/sshd
  172 root      2200 S    -bash
  174 root      1592 S    ps –ef
root@Moxa:~#
```

Setting the Run-level

In this section, we outline the steps you should take to set the Linux run-level and execute requests. Use the following command to enable or disable settings:

```
192.168.3.127 - PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common  S25nfs-user-server S99showreadyled
S20snmpd    S55ssh
S24pcmcia  S99rmnologin
root@Moxa:/etc/rc.d/rc3.d#
```

#cd /etc/rc.d/init.d

Edit a shell script to execute **/root/tcps2-release** and save to **tcps2** as an example.

#cd /etc/rc.d/rc3.d
#ln –s /etc/rc.d/init.d/tcps2 S60tcps2

SxxRUNFILE stands for
S: start the run file while Linux boots up.

xx: a number between 00-99. The smaller number has a higher priority.
RUNFILE: the file name.

```
  192.168.3.127 - PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common  S25nfs-user-server S99showreadyled
S20snmpd    S55ssh
S24pcmcia  S99rmnologin
root@Moxa:/ect/rc.d/rc3.d# ln –s /root/tcps2-release S60tcps2
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common  S25nfs-user-server S99rmnologin
S20snmpd    S55ssh     S99showreadyled
S24pcmcia     S60tcps2
root@Moxa:/etc/rc.d/rc3.d#
```

```
#cd  /etc/rc.d/rc6.d
#ln  –s  /etc/rc.d/init.d/tcps2  K30tcps2
```

KxxRUNFILE stands for
K: start the run file while Linux shuts down or halts.
xx: a number between 00-99. The smaller number has a higher priority.
RUNFILE: is the file name.

For removing the daemon, you can remove the run file from **/etc/rc.d/rc3.d** by using the following command:

```
#rm –f /etc/rc.d/rc3.d/S60tcps2
```

Adjusting the System Time

Setting the Time Manually

The DA-660 has two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the DA-660 hardware. Use the **#date** command to query the current system time or set a new system time. Use **#hwclock** to query the current RTC time or set a new RTC time.

Use the following command to query the system time:

```
#date
```

Use the following command to query the RTC time:

```
#hwclock
```

Use the following command to set the system time:

```
#date MMDDhhmmYYYY
```

MM = Month
DD = Date
hhmm = hour and minute
YYYY = Year

Use the following command to set the RTC time:

```
#hwclock –w
```

Write current system time to RTC:

The following figure illustrates how to update the system time and set the RTC time.

```
192.168.3.127 - PuTTY
root@Moxa:~# date
Fri Jun 23 23:30:31 CST 2000
root@Moxa:~# hwclock
Fri Jun 23 23:30:35 2000  -0.557748 seconds
root@Moxa:~# date 120910002004
Thu Dec 9 10:00:00 CST 2004
root@Moxa:~# hwclock –w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:01:07 CST 2004
Thu Dec  9 10:01:08 2004  -0.933547 seconds
root@Moxa:~#
```

NTP Client

The DA-660 has a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <this client utility>** to update the system time.

```
#ntpdate time.stdtime.gov.tw
#hwclock –w
```

Visit http://www.ntp.org for more information about NTP and NTP server addresses.

```
10.120.53.100 - PuTTY
root@Moxa:~# date ; hwclock
Sat Jan  1 00:00:36 CST 2000
Sat Jan  1 00:00:37 2000  -0.772941 seconds
root@Moxa:~# ntpdate time.stdtime.gov.tw
 9 Dec 10:58:53 ntpdate[207]: step time server 220.130.158.52 offset 155905087.984256
sec
root@Moxa:~# hwclock –w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:59:11 CST 2004
Thu Dec  9 10:59:12 2004  -0.844076 seconds
root@Moxa:~#
```

NOTE Before using the NTP client utility, check your IP and DNS settings to make sure that an Internet connection is available. Refer to Chapter 2 for instructions on how to configure the Ethernet interface, and Chapter 4 for information on setting up the DNS.

Updating the Time Automatically

In this subsection we demonstrate the usage of a shell script to update the time automatically.

Example shell script to update the system time periodically

```
#!/bin/sh
ntpdate time.nist.gov  # You can use the time server's ip address or domain
                       # name directly. If you use domain name, you must
                       # enable the domain client on the system by updating
                       # /etc/resolv.conf file.
hwclock –systohc
sleep 100    # Updates every 100 seconds. The min. time is 100 seconds. Change
             # 100 to a larger number to update RTC less often.
```

Save the shell script using any file name (e.g., **fixtime**).

How to run the shell script automatically when the kernel boots up

Copy the example shell script **fixtime** to directory **/etc/init.d**, and then use **chmod 755 fixtime** to make the shell script executable. Next, use the vi editor to edit the file **/etc/inittab**. Add the following line to the bottom of the file:

```
ntp : 2345 : respawn : /etc/init.d/fixtime
```

Use the command **#init q** to re-init the kernel.

Cron—Daemon to Execute Scheduled Commands

Start Cron from the directory /**etc/rc.d/rc. local**. The cron process will return immediately, so you do not need to start it with '&' to run in the background.

The Cron daemon will search /etc/cron.d/crontab for crontab files, which are named after accounts in /etc/passwd.

Cron wakes up every minute, and checks each command to see if it should be run in the current minute. When executing commands, output is mailed to the owner of the crontab (or to the user named in the MAILTO environment variable in the crontab, if such a user exists).

Modify the file /etc/cron.d/crontab to set up your scheduled applications. Crontab files have the following format:

mm	h	dom	mon	dow	user	command
month	hour	date	month	week	user	command
0-59	0-23	1-31	1-12	0-6 (0 is Sunday)		

The following example demonstrates how to use Cron.

How to use cron to update the system time and RTC time every day at 8:00.

STEP1: Write a shell script named fixtime.sh and save it to /home/.

```
#!/bin/sh
ntpdate time.nist.gov
hwclock –-systohc
exit 0
```

STEP2: Change mode of fixtime.sh

```
#chmod 755 fixtime.sh
```

STEP3: Modify /etc/cron.d/crontab file to run fixtime.sh at 8:00 every day.

Add the following line to the end of crontab:

```
* 8 * * * root /home/fixtime.sh
```

STEP4: Enable the cron daemon manually.

```
#/etc/init.d/cron start
```

STEP5: Enable cron when the system boots up.

Add the following line in the file /etc/init.d/rc.local

```
#/etc/init.d/cron start
```

Timezone Setting

Two methods are available for configuring the timezone setting.

1. Using the TZ variable

   ```
   TZ=standardHH[:MM[:SS]][daylight[HH[:MM[:SS]]][,startdate[/starttime],
   enddate[/endtime]]]
   ```

   The time kept by the local machine should be a universal standard representation, such as Greenwich Mean Time (GMT) or Universal Time Coordinated (UTC), which we refer to collectively as the "universal reference time." For personal computers that do not share data

across time zones, the local time is an adequate standard. To support a universal standard, all MKS utilities assume that times stored in the file system and returned by the operating system are stored in the universal reference time, and then translated to local times. The mapping from the universal reference time to local time is specified by the TZ (time zone) environment variable. If left undefined, the TZ variable defaults to the current time zone setting of your operating system.

Here are some possible settings for the North American Eastern time zone:

```
TZ=EST5EDT
TZ=EST0EDT
TZ=EST0
```

In the first case, the reference time is GMT and the stored time values are correct worldwide. A simple change of the TZ variable prints local time correctly anywhere. In the second case, the reference time is Eastern Standard Time and the only conversion performed is for Daylight Savings Time. For this reason, there is no need to adjust the hardware clock for Daylight Savings Time twice a year. In the third case, the reference time is always the time reported. This is suggested if the hardware clock on your machine automatically adjusts for Daylight Savings Time or you insist on manually resetting the hardware time twice a year.

Other examples include:

```
TZ=NST3:30NDT2:00
TZ=MSEZ-1
```

The first applies to Newfoundland, whereas the second works in most of Western Europe.

Here are some time zone scenarios that involve Daylight Savings Time specifications:

```
TZ=PST0PDT-1
TZ=ACST-09:30ACDT-10:30,M10.5.0/2:00,M3.5.0/2:00
```

The first scenario shows the TZ of a person in Seattle who stores local time on a PC, but does not adjust the clock to agree with Daylight Savings Time. The stated time zone precedes the machine clock time by one hour when Daylight Savings Time is in effect.

The second scenario shows the TZ set by a person in Australia who sets a PC clock to UTC and never adjusts it. The machine clock precedes UTC by 9.5 hours when Daylight Savings Time is not in effect, and by 10.5 hours when it is in effect. Daylight Savings Time is in effect from 2:00 am on the last Sunday in October until 2:00 am on the last Sunday in March.

To cause the timezone setting to be activated after the DA-660 reboots, add the line

```
export TZ=your_timezone_setting
```

to the file **/etc/rc.d/rc.local**.

2. How to use the file **/etc/localtime**.

   The local timezone is stored in **/etc/localtime** and is used by GNU Library for C (glibc) if the TZ environment variable is not set. This file is either a copy of the **/usr/share/zoneinfo/** tree or a symbolic link to it.

   The DA-660 does not provide **/use/share/zoneinfo/** files, so you will need to copy a time zone information file to the DA-660 and write over the original local time file.

   - The **/usr/share/zoneinfo** folder on a PC that is running standard Linux contains several time zone information files.
   - Copy the time zone file that you want to use to the DA-660, and write over the original **/etc/localtime** file.
   - Type the date to check if the new time zone appears.

4

Managing Communications

In this chapter, we explain how to configure the DA-660's various communication functions.

The following topics are covered in this chapter:

- **Telnet / FTP**
- **DNS**
- **Web Service—Apache**
- **IPTABLES**
- **NAT**
  - NAT Example
  - Enabling NAT at Bootup
- **Dial-up Service—PPP**
- **PPPoE**
- **NFS (Network File System)**
  - Setting up the DA-660 as a NFS Server
  - Setting up the DA-660 as a NFS Client
- **Mail**
- **SNMP**
- **OpenVPN**

Telnet / FTP

In addition to supporting Telnet client/server and FTP client/server, the DA-660 also supports SSH and SFTP client/server. To enable or disable the Telnet/FTP server, you first need to edit the file **/etc/inetd.conf**.

Enabling the Telnet/FTP server

The following example shows the default content of the file **/etc/inetd.conf**. The default is to enable the Telnet/ftp server:

```
discard dgram udp wait root /bin/discard
discard stream tcp nowait root /bin/discard
telnet stream tcp nowait root /bin/telnetd
ftp stream tcp nowait root /bin/ftpd -l
```

Disabling the Telnet/FTP server

Disable the daemon by placing a '#' in front of the first character of the row to comment out the line.

```
#telnet stream tcp nowait root /bin/telnetd
#ftp stream tcp nowait root /bin/ftpd -l
```

DNS

The DA-660 supports the DNS client (but not the DNS server). To set up the DNS client, you need to edit three configuration files: **/etc/hosts, /etc/resolv.conf,** and **/etc/nsswitch.conf**.

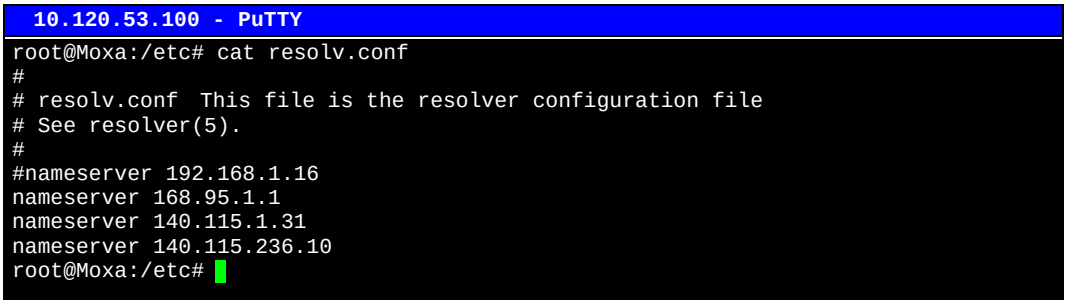
/etc/hosts

This is the first file that the Linux system reads to resolve the host name and IP address.

/etc/resolv. conf

This is the most important file that you need to edit when using DNS for the other programs. For example, before using **#ntpdate time.nist.goc** to update the system time, you will need to add the DNS server address to the file. Ask your network administrator for the DNS server address to use. The DNS server's IP address is specified with the "nameserver" command. For example, add the following line to **/etc/resolv.conf** if the DNS server's IP address is 168.95.1.1:

```
nameserver 168.95.1.10
```



/etc/nsswitch.conf

This file defines the sequence of files to be read to resolve the IP address - **/etc/hosts** or **/etc/resolv.conf**.

Web Service—Apache

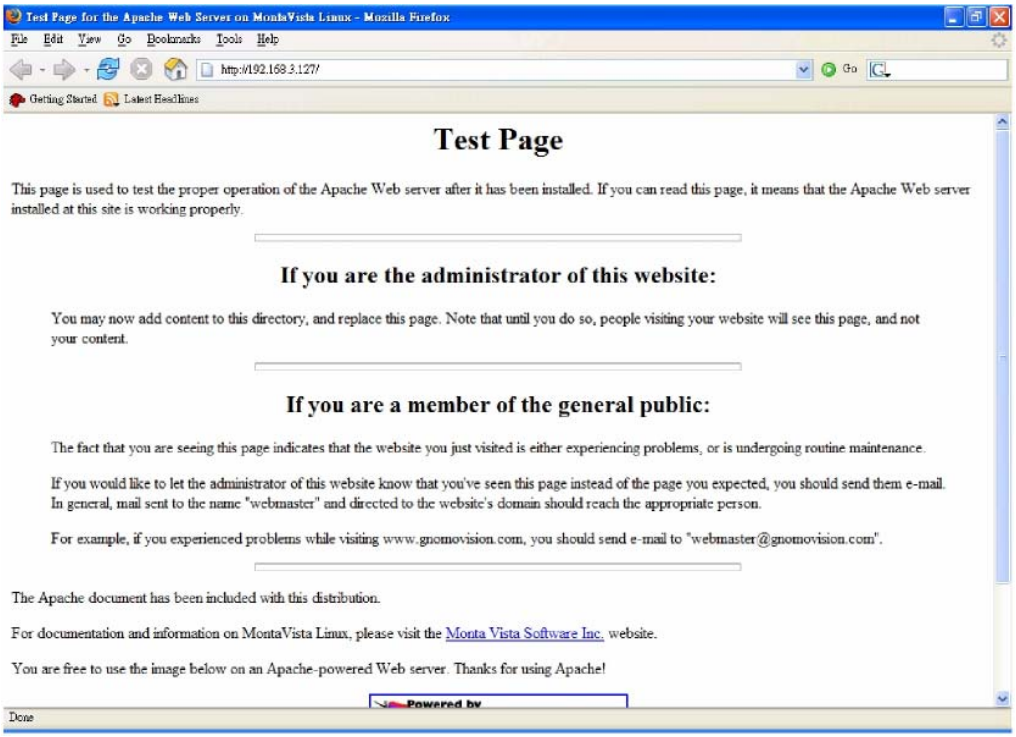
The Apache web server's main configuration file is **/etc/apache/httpd.conf**, with the default homepage located at **/usr/www/html/index.html**. Save your own homepage to the following directory:

```
/usr/www/html/
```

Save your CGI page to the following directory:

```
/usr/www/cgi-bin/
```

Before you modify the homepage, use a browser (such as Microsoft Internet Explore or Mozilla Firefox) from your PC to test if the Apache Web Server is working. Type the LAN1 IP address in the browser's address box to open the homepage. E.g., if the default IP address is still active, type **http://192.168.3.127** in the address box.


Test Page for the Apache Web Server on MontaVista Linux - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.3.127/
Getting Started Latest Headlines
Test Page
This page is used to test the proper operation of the Apache Web server after it has been installed. If you can read this page, it means that the Apache Web server installed at this site is working properly.
If you are the administrator of this website:
You may now add content to this directory, and replace this page. Note that until you do so, people visiting your website will see this page, and not your content.
If you are a member of the general public:
The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.
If you would like to let the administrator of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.
For example, if you experienced problems while visiting www.gnomovision.com, you should send e-mail to "webmaster@gnomovision.com".
The Apache document has been included with this distribution.
For documentation and information on MontaVista Linux, please visit the Monta Vista Software Inc. website.
You are free to use the image below on an Apache-powered Web server. Thanks for using Apache!
Done

To open the default CGI page, type **http://192.168.3.127/cgi-bin/printenv** in your browser's address box.

To open the default CGI test script report page, type **http://192.168.3.127/cgi-bin/test-cgi** in your browser's address box.

NOTE The CGI function is enabled by default. If you want to disable the function, modify the file **/etc/apache/conf/httpd.conf**. When you develop your own CGI application, make sure your CGI file is executable.

```
192.168.3.127 - PuTTY
root@Moxa:/usr/www/cgi-bin# ls -al
drwxr-xr-x   2 root  root       0 Aug 24 1999
drwxr-xr-x   5 root  root       0 Nov  5 16:16
-rwxr-xr-x   1 root  root     268 Dec 19 2002 printenv
-rwxr-xr-x   1 root  root     757 Aug 24 1999 test-cgi
root@Moxa:/usr/www/cgi-bin#
```

IPTABLES

IPTABLES is an administrative tool for setting up, maintaining, and inspecting the Linux kernel's IP packet filter rule tables. Several different tables are defined, with each table containing built-in chains and user-defined chains.

Each chain is a list of rules that apply to a certain type of packet. Each rule specifies what needs to be done with a matching packet. A rule (such as a jump to a user-defined chain in the same table) is called a **target**.

The DA-660 supports 3 types of IPTABLES table: **Filter** tables, **NAT** tables, and **Mangle** tables:

A. Filter Table—includes three chains:

INPUT chain
OUTPUT chain
FORWARD chain

B. NAT Table—includes three chains:

PREROUTING chain—transfers the destination IP address (DNAT)
POSTROUTING chain—works after the routing process and before the Ethernet device process to transfer the source IP address (SNAT)
OUTPUT chain—produces local packets

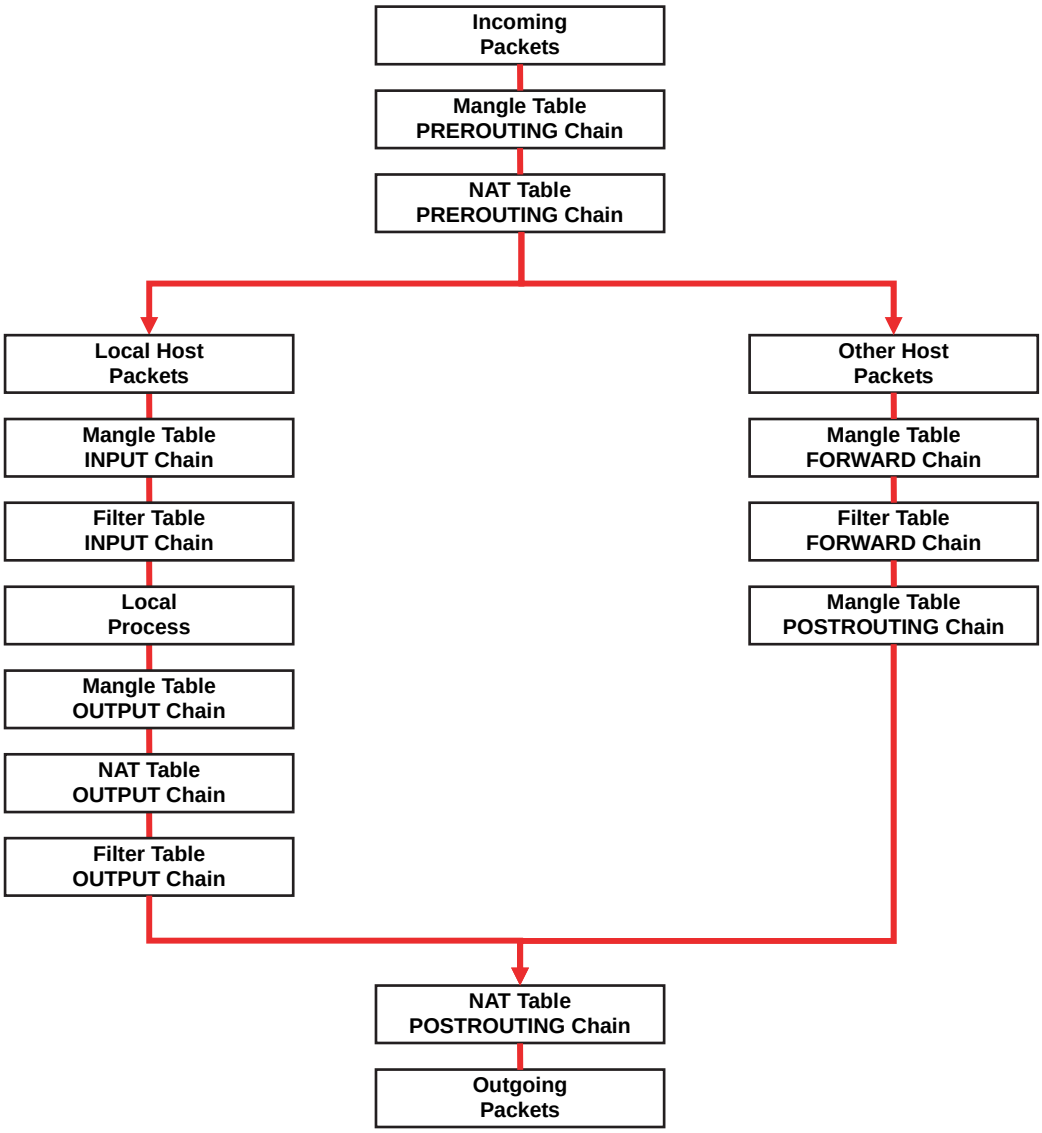
sub-tables

Source NAT (SNAT)—changes the first source packet IP address
Destination NAT (DNAT)—changes the first destination packet IP address
MASQUERADE—a special form for SNAT. If one host can connect to internet, then other computers that connect to this host can connect to the Internet even if these computers does not have an actual IP address.
REDIRECT—a special form of DNAT that re-sends packets to a local host independent of the destination IP address.

C. Mangle Table—includes two chains

PREROUTING chain—pre-processes packets before the routing process.
OUTPUT chain—processes packets after the routing process.
It has three extensions—TTL, MARK, TOS.

The following figure shows the IPTABLES hierarchy.



The DA-660 supports the following sub-modules. Be sure to use the module that matches your application.

ip_conntrack	ipt_MARK	ipt_ah	ipt_state
ip_conntrack_ftp	ipt_MASQUERADE	ipt_esp	ipt_tcpmss
ipt_conntrack_irc	ipt_MIRROT	ipt_length	ipt_tos
ip_nat_ftp	ipt_REDIRECT	ipt_limit	ipt_ttl
ip_nat_irc	ipt_REJECT	ipt_mac	ipt_unclean
ip_nat_snmp_basic	ipt_TCPMSS	ipt_mark	
ip_queue	ipt_TOS	ipt_multiport	
ipt_LOG	ipt_ULOG	ipt_owner	

NOTE The DA-660 does NOT support IPV6 and ipchains.

The basic syntax to enable and load an IPTABLES module is as follows:

```
#lsmod
#modprobe ip_tables
#modprobe iptable_filter
```

Use lsmod to check if the ip_tables module has already been loaded in the DA-660. Use **modprobe** to insert and enable the module.

Use the following command to load the modules (**iptable_filter, iptable_mangle, iptable_nat**):

```
#modprobe iptable_filter
```

Use **iptables, iptables-restore,** and **iptables-save** to maintain the database.

NOTE IPTABLES acts as a packet filter or NAT. Take care when setting up the IPTABLES rules. If the rules are not correct, remote hosts that connect via a LAN or PPP may be denied access. We recommend using the Serial Console to set up the IPTABLES.

Click on the following links for more information about IPTABLES.

http://www.linuxguruz.com/iptables/
http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html

Since the IPTABLES command is very complex, to illustrate the IPTABLES syntax we have divided our discussion of the various rules into three categories: **Observe and erase chain rules, Define policy rules,** and **Append or delete rules**.

Observe and erase chain rules

Usage:

```
# iptables [-t tables] [-L] [-n]
```

-t tables: Table to manipulate (default: 'filter'); example: nat or filter.
-L [chain]: List List all rules in selected chains. If no chain is selected, all chains are listed.
-n: Numeric output of addresses and ports.

```
# iptables [-t tables] [-FXZ]
```

-F: Flush the selected chain (all the chains in the table if none is listed).
-X: Delete the specified user-defined chain.
-Z: Set the packet and byte counters in all chains to zero.

Examples:

```
# iptables -L -n
```

In this example, since we do not use the -t parameter, the system uses the default 'filter' table. Three chains are included: INPUT, OUTPUT, and FORWARD. INPUT chains are accepted automatically, and all connections are accepted without being filtered.

```
#iptables –F
#iptables –X
#iptables -Z
```

Define policy for chain rules

Usage:

```
# iptables [-t tables] [-P] [INPUT, OUTPUT, FORWARD, PREROUTING, OUTPUT, POSTROUTING]
[ACCEPT, DROP]
```

-P: Set the policy for the chain to the given target.
INPUT: For packets coming into the DA-660.
OUTPUT: For locally-generated packets.
FORWARD: For packets routed out through the DA-660.
PREROUTING: To alter packets as soon as they come in.
POSTROUTING: To alter packets as they are about to be sent out.

Examples:

```
#iptables –P INPUT DROP
#iptables –P OUTPUT ACCEPT
#iptables –P FORWARD ACCEPT
#iptables –t nat –P PREROUTING ACCEPT
#iptables –t nat –P OUTPUT ACCEPT
#iptables -t nat –P POSTROUTING ACCEPT
```

In this example, the policy accepts outgoing packets and denies incoming packets.

Append or delete rules

Usage:

```
# iptables [-t table] [-AI] [INPUT, OUTPUT, FORWARD] [-io interface] [-p tcp, udp, icmp,
all] [-s IP/network] [--sport ports] [-d IP/network] [--dport ports] –j [ACCEPT. DROP]
```

-A: Append one or more rules to the end of the selected chain.
-I: Insert one or more rules in the selected chain as the given rule number.
-i: Name of an interface via which a packet is going to be received.
-o: Name of an interface via which a packet is going to be sent.
-p: The protocol of the rule or of the packet to check.
-s: Source address (network name, host name, network IP address, or plain IP address).
--sport: Source port number.
-d: Destination address.
--dport: Destination port number.
-j: Jump target. Specifies the target of the rules; i.e., how to handle matched packets. For example, ACCEPT the packet, DROP the packet, or LOG the packet.

Examples:

Example 1: Accept all packets from lo interface.
```
# iptables –A INPUT –i lo –j ACCEPT
```

Example 2: Accept TCP packets from 192.168.0.1.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.1 –j ACCEPT
```

Example 3: Accept TCP packets from Class C network 192.168.1.0/24.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.0/24 –j ACCEPT
```

Example 4: Drop TCP packets from 192.168.1.25.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.25 –j DROP
```

Example 5: Drop TCP packets addressed for port 21.
```
# iptables –A INPUT –i eth0 –p tcp --dport 21 –j DROP
```

Example 6: Accept TCP packets from 192.168.0.24 to DA-660's port 137, 138, 139
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.24 --dport 137:139 –j ACCEPT
```

Example 7: Log TCP packets that visit DA-660's port 25
```
# iptables –A INPUT –i eth0 –p tcp --dport 25 –j LOG
```

Example 8: Drop all packets from MAC address 01:02:03:04:05:06
```
# iptables –A INPUT –i eth0 –p all –m mac –mac-source 01:02:03:04:05:06 –j DROP
```

NOTE: In Example 8, remember to issue the command **#modprobe ipt_mac** first to load module **ipt_mac**.

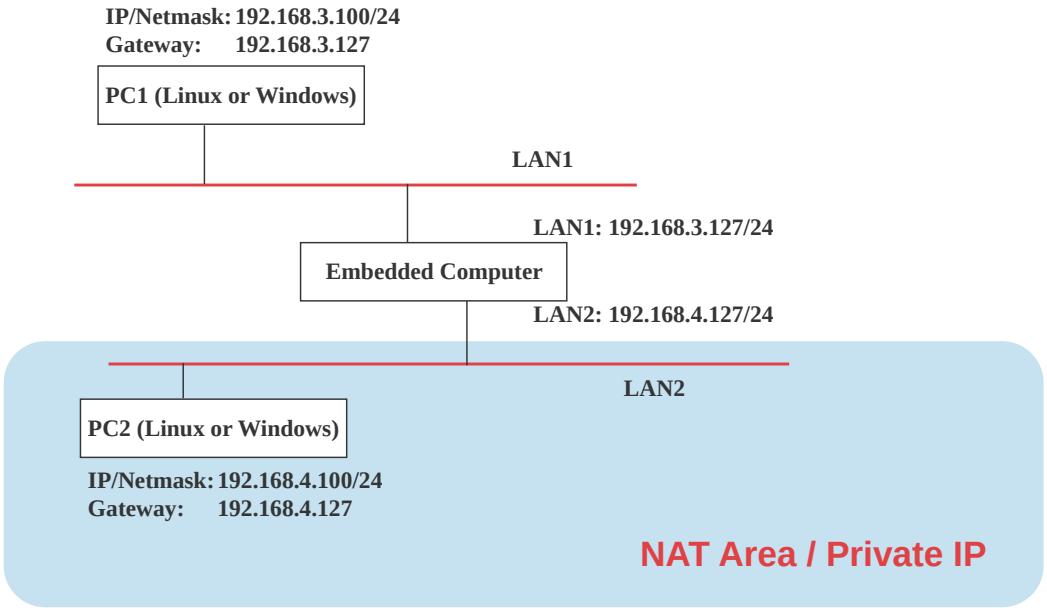
NAT

NAT (Network Address Translation) protocol translates IP addresses used on one network to different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, the DA-660 connects several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and un-maps the global IP addresses on incoming packets back into local IP addresses.

NOTE Click on the following link for more information about iptables and NAT:
http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html

NAT Example

The IP address of LAN1 is changed to 192.168.3.127 (you will need to load the module ipt_MASQUERADE):




```
1. echo "1" > /proc/sys/net/ipv4/ip_forward
2. modprobe ip_tables
3. modprobe ip_conntrack
4. modprobe iptable_nat
5. modprobe ipt_MASQUERADE
6. iptables -t nat -A POSTROUTING -o ixp0 -j MASQUERADE
```

Enabling NAT at Bootup

In the most real world situations, you will want to use a simple shell script to enable NAT when the DA-660 boots up. The following script is an example.

```
#!/bin/bash
# If you put this shell script in the /home/nat.sh
# Remember to chmod 744 /home/nat.sh
# Edit the rc.local file to make this shell startup automatically.
# vi /etc/rc.d/rc.local
# Add a line in the end of rc.local /home/nat.sh
EXIF='eth0' #This is an external interface for setting up a valid IP address.
EXNET='192.168.4.0/24' #This is an internal network address.
# Step 1. Insert modules.
# Here 2> /dev/null means the standard error messages will be dump to null device.
modprobe ip_tables 2> /dev/null
modprobe ip_nat_ftp  2> /dev/null
modprobe ip_nat_irc  2> /dev/null
modprobe ip_conntrack  2> /dev/null
modprobe ip_conntrack_ftp  2> /dev/null
modprobe ip_conntrack_irc  2> /dev/null
# Step 2. Define variables, enable routing and erase default rules.
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
export PATH
echo "1" > /proc/sys/net/ipv4/ip_forward
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -Z
/sbin/iptables -F -t nat
/sbin/iptables -X -t nat
/sbin/iptables -Z -t nat
/sbin/iptables -P INPUT ACCEPT
/sbin/iptables -P OUTPUT ACCEPT
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -t nat -P PREROUTING ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P OUTPUT ACCEPT
# Step 3. Enable IP masquerade.
```

Dial-up Service—PPP

PPP (Point to Point Protocol) is used to run IP (Internet Protocol) and other network protocols over a serial link. PPP can be used for direct serial connections (using a null-modem cable) over a Telnet link, and links established using a modem over a telephone line.

Modem / PPP access is almost identical to connecting directly to a network through the DA-660's Ethernet port. Since PPP is a peer-to-peer system, the DA-660 can also use PPP to link two networks (or a local network to the Internet) to create a Wide Area Network (WAN).

NOTE Click the following links for more information about PPP:
http://tldp.org/HOWTO/PPP-HOWTO/index.html
http://axion.physics.ubc.ca/ppp-linux.html

The pppd daemon is used to connect to a PPP server from a Linux system. For detailed information about pppd, see the man page.

Example 1: Connecting to a PPP server over a simple dial-up connection

The following command is used to connect to a PPP server using a modem. Use this command for old ppp servers that prompt for a login name (replace *username* with the correct name) and password (replace *password* with the correct password). Note that *debug* and *defaultroute 192.1.1.17* are optional.

```
#pppd connect 'chat -v " " ATDT5551212 CONNECT" " ogin: username word: password'
/dev/ttyM0 115200 debug crtscts modem defaultroute
```

If the PPP server does not prompt for the username and password, the command should be entered as follows. Replace *username* with the correct username and replace *password* with the correct password.

```
#pppd connect 'chat -v " " ATDT5551212 CONNECT" "' user username password password
/dev/ttyM0 115200 crtscts modem
```

The pppd options are as follows:

connect 'chat etc...'

This option provides the command to contact the PPP server. The 'chat' program is used to dial a remote computer. The entire command is enclosed in single quotes since pppd expects a one-word argument for the 'connect' option. The options for 'chat' are given below:

-v
verbose mode; log what we do to syslog

" "
Double quotes—do not wait for a prompt, but instead do ... (note that you must include a space after the second quotation mark)

ATDT5551212
Dial the modem, and then ...

CONNECT
Wait for an answer.

" "
Send a return (null text followed by the usual return)

ogin: username word: password
Log in with username and password.

Refer to the chat man page, chat.8, for more information about the chat utility.

/dev/
Specify the callout serial port.

115200
The baudrate.

debug
Log status in syslog.

crtscts
Use hardware flow control between computer and modem (at a baudrate of 115200, this is a must).

modem
Indicates that this is a modem device; pppd will hang up the phone before and after making the call.

defaultroute

Once the PPP link is established, make it the default route; if you have a PPP link to the Internet, this is probably what you want.

192.1.1.17

This is a degenerate case of a general option of the form x.x.x.x:y.y.y.y. Here x.x.x.x is the local IP address and y.y.y.y is the IP address of the remote end of the PPP connection. If this option is not specified, or if just one side is specified, then x.x.x.x defaults to the IP address associated with the local machine's hostname (located in /etc/hosts), and y.y.y.y is determined by the remote machine.

Example 2: Connecting to a PPP server over a hard-wired link

If a username and password are not required, use the following command (note that noipdefault is optional):

#pppd connect 'chat –v" " " " ' *noipdefault* /dev/ttyM0 19200 crtscts

If a username and password are required, use the following command (note that noipdefault is optional, and root is both the username and password):

#pppd connect 'chat –v" " " " ' user *root* password *root* *noipdefault* /dev/ttyM0 19200 crtscts

Checking the connection

Once you have set up a PPP connection, there are some steps that you can take to test the connection. First, type:

/sbin/ifconfig

(The folder **ifconfig** may be located elsewhere, depending on your distribution.) You should be able to see all the network interfaces that are UP. ppp0 should be one of them, and you should recognize the first IP address as your own, and the "P-t-P address" (or point-to-point address) the address of your server. The output is similar to the following:

```
lo      Link encap Local Loopback
        inet addr 127.0.0.1     Bcast 127.255.255.255     Mask 255.0.0.0
        UP LOOPBACK RUNNING     MTU 2000     Metric 1
        RX packets 0 errors 0 dropped 0 overrun 0

ppp0    Link encap Point-to-Point Protocol
        inet addr 192.76.32.3     P-t-P 129.67.1.165     Mask 255.255.255.0
        UP POINTOPOINT RUNNING    MTU 1500     Metric 1
        RX packets 33 errors 0 dropped 0 overrun 0
        TX packets 42 errors 0 dropped 0 overrun 0
```

Now, type:

ping z.z.z.z

where z.z.z.z is the address of your name server. This should work. The output is similar to the following:

```
waddington:~$p ping 129.67.1.165
PING 129.67.1.165 (129.67.1.165): 56 data bytes
64 bytes from 129.67.1.165: icmp_seq=0 ttl=225 time=268 ms
64 bytes from 129.67.1.165: icmp_seq=1 ttl=225 time=247 ms
64 bytes from 129.67.1.165: icmp_seq=2 ttl=225 time=266 ms
^C
--- 129.67.1.165 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
```

round-trip min/avg/max = 247/260/268 ms
waddington:~$

Try typing:

```
netstat –nr
```

This should show three routes similar to the following:

Kernel routing table

Destination iface	Gateway	Genmask	Flags	Metric	Ref	Use
129.67.1.165 ppp0	0.0.0.0	255.255.255.255	UH	0	0	6
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0 lo
0.0.0.0 ppp0	129.67.1.165	0.0.0.0	UG	0	0	6298

If your output looks similar but does not have the destination 0.0.0.0 line (which refers to the default route used for connections), you may have run pppd without the 'defaultroute' option. At this point you can try using Telnet, FTP, or finger, bearing in mind that you will have to use numeric IP addresses unless you've set up /etc/resolv.conf correctly.

Setting up a Machine for Incoming PPP Connections

This first example applies to using a modem, and requiring authorization with a username and password.

```
pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2 login auth
```

You should also add the following line to the file **/etc/ppp/pap-secrets**:

```
*  *    ""   *
```

The first star (*) lets everyone login. The second star (*) lets every host connect. The pair of double quotation marks ("") is to use the file **/etc/passwd** to check the password. The last star (*) is to let any IP connect.

The following example does not check the username and password:

```
pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2
```

PPPoE

1. Connect the DA-660's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.
2. Login to the DA-660 as the root user.
3. Edit the file **/etc/ppp/chap-secrets** and add the following:
   "username@hinet.net" * "password" *

```
# Secrets for authentication using CHAP
# client      server  secret              IP addresses
"username@hinet.net"   *      "password"   *
```

"username@hinet.net" is the username obtained from the ISP to log in to the ISP account.
"password" is the corresponding password for the account.

4. Edit the file **/etc/ppp/pap-secrets** and add the following:
   "username@hinet.net" * "password" *

```
support hostname       "*"     -
stats   hostname       "*"     -

# OUTBOUND connections
# ATTENTION: The definitions here can allow users to login without a
# package already provides this option; make sure you don't change that.

# INBOUND connections

# Every regular user can use PPP and has to use passwords from /etc/passwd
*       hostname       ""      *
"username@hinet.net"     *    "password"    *

# PPPOE user example, if you want to use it, you need to unmark it and modify it
#"username@hinet.net"   *      "password"     *

# UserIDs that cannot use PPP at all. Check your /etc/passwd and add any
# other accounts that should not be able to use pppd!
guest   hostname       "*"     -
master  hostname       "*"     -
root    hostname       "*"     -
support hostname       "*"     -
stats   hostname       "*"     -
```

   "username@hinet.net" is the username obtained from the ISP to log in to the ISP account. **"password"** is the corresponding password for the account.

5. Edit the file **/etc/ppp/options** and add the following line:
   plugin pppoe.so

```
# Wait for up n milliseconds after the connect script finishes for a valid
# PPP packet from the peer.  At the end of this time, or when a valid PPP
# packet is received from the peer, pppd will commence negotiation by
# sending its first LCP packet.  The default value is 1000 (1 second).
# This wait period only applies if the connect or pty option is used.
#connect-delay <n>

# Load the pppoe plugin
plugin pppoe.so

# ---<End of File>---
```

6. Add one of two files: **/etc/ppp/options.eth0** or **/etc/ppp/options.eth1**. The choice depends on is the LAN that is connected to the ADSL modem. If you use LAN1 to connect to the ADSL modem, then add **/etc/ppp/options.eth0**. If you use LAN2 to connect to the ADSL modem, then add **/etc/ppp/options.eth1**. The file context is shown below:

   The content of the file is as follows:

```
name username@hinet.net
mtu 1492
mru 1492
defaultroute
noipdefault
```

   Type your username (the one you set in the **/etc/ppp/pap-secrets** and **/etc/ppp/chap-secrets** files) after the "name" option. You may add other options as desired.

7. Set up DNS
   If you are using DNS servers supplied by your ISP, edit the file
   /etc/resolv.conf by adding the following lines of code:
   `nameserver ip_addr_of_first_dns_server`

```
nameserver ip_addr_of_second_dns_server
```

For example:

```
nameserver 168..95.1.1
nameserver 139.175.10.20
```

8. Use the following command to create a pppoe connection:
   ```
   pppd eth0
   ```
   The eth0 is what is connected to the ADSL modem LAN port. The example above uses LAN1. To use LAN2, type:
   ```
   pppd eth1
   ```
9. Type **ifconfig ppp0** to check if the connection is OK or has failed. If the connection is OK, you will see information about the ppp0 setting for the IP address. Use ping to test the IP.
10. If you want to disconnect it, use the kill command to kill the pppd process.

NFS (Network File System)

The Network File System (NFS) is used to mount a disk partition on a remote machine, as if it were on a local hard drive, allowing fast, seamless sharing of files across a network. NFS allows users to develop applications for the DA-660, without worrying about the amount of disk space that will be available. The DA-660 supports the NFS protocol for both client and server.

NOTE Click the following links for more information about NFS:
http://www.tldp.org/HOWTO/NFS-HOWTO/index.html
http://nfs.sourceforge.net/nfs-howto/client.html
http://nfs.sourceforge.net/nfs-howto/server.html

Setting up the DA-660 as a NFS Server

By default, the DA-660 enables the NFS service **/etc/init.d/nfs-user-server**. The service link file **S25nfs-user-server** is located in the directory **/rc.d/rc2.d-rc5.d**.

Edit the NFS server configuration file **/etc/exports** to set up the remote host (NFS client) list and access rights for a specific directory. The file formats are shown below:

```
#vi /etc/exports
```

File Format:

```
directory machine1(option11,option12) machine2(option21,option22)
```

directory
The directory that will be shared with the NFS Client.

machine1 and machine2
Client machines that will have access to the directory. A machine can be listed by its DNS address or IP address (e.g., machine.company.com or 192.168.0.8).

optionxx
The option list for a machine describes the kind of access the machine will have. Important options are:

ro
Read only. This is the default.

rw
Readable and Writeable.

no_root_squash

If **no_root_squash** is selected, then the root on the client machine will have the same level of access to files on the system as the root on the server. This can have serious security implications, although it may be necessary if you want to do administrative work on the client machine that involves the exported directories. You should only specify this option when you have a good reason.

root_squash

Any file request made by the user root on the client machine is treated as if it is made by user nobody on the server. (Exactly which UID the request is mapped to depends on the UID of user **nobody** on the server, not the client.)

sync

Sync data to memory and flash disk.

async

The async option instructs the server to lie to the client, telling the client that all data has been written to the stable storage.

Example 1

```
/tmp  *(rw,no_root_squash)
```

In this example, the DA-660 shares the **/tmp** directory with everyone, and gives everyone both read and write authority. The root user on the client machine will have the same level of access to files on the system as the root on the server.

Example 2

```
/home/public 192.168.0.0/24(rw) *(ro)
```

In this example, the DA-660 shares the directory **/home/public** to a local network 192.168.0.0/24, with read and write authority. Other NFS clients can just read **/home/public**; they do not have write authority.

Example 3

```
/home/test  192.168.3.100(rw)
```

In this example, the DA-660 shares the directory **/home/test** to an NFS Client 192.168.3.100, with both read and write authority.

NOTE After editing the NFS Server configuration file, remember to use the following command to restart and activate the NFS server.

```
/etc/init.d/nfs-user-server restart
```

Setting up the DA-660 as a NFS Client

Use the following procedure is used to mount a remote NFS Server.

1. Scan the NFS Server's shared directory.
2. Establish a mount point on the NFS Client site.
3. Mount the remote directory to a local directory.

Step 1:

```
#showmount –e HOST
```

showmount:	`Show the mount information for an NFS Server.`
-e:	`Show the NFS Server's export list.`
HOST:	`IP address or DNS address.`

Steps 2 & 3:

```
#mkdir –p /home/nfs/public
```

```
#mount –t nfs NFS_Server(IP):/directory /mount/point
```

Example:

```
#mount –t nfs 192.168.3.100/home/public  /home/nfs/public
```

Mail

smtpclient is a minimal SMTP client that takes an email message body and passes it on to an SMTP server. It is suitable for applications that use email to send alert messages or important logs to a specific user.

NOTE Click on the following link for more information about smtpclient:
http://www.engelschall.com/sw/smtpclient/

To send an email message, use the 'smtpclient' utility that uses SMTP protocol. Type **#smtpclient –help** to see the help message.

Example:

```
smtpclient –s test –f sender@company.com –S IP_address receiver@company.com
< mail-body-message
```

- **-s:** The mail subject.
- **-f:** Sender's mail address
- **-S:** SMTP server IP address

The last mail address **receiver@company.com** is the receiver's e-mail address.
mail-body-message is the mail content. The last line of the body of the message should contain ONLY the period '.' character.

You will need to add your hostname to the file **/etc/hosts**.

SNMP

The DA-660 has built-in SNMP V1 (Simple Network Management Protocol) agent software. It supports RFC1317, RS-232 like groups, and RFC 1213 MIB-II.

The following simple example allows you to use an SNMP browser on the host site to query the DA-660, which is the SNMP agent. The DA-660 will respond.

```
***** SNMP QUERY STARTED *****
1: sysDescr.0 (octet string) Linux Moxa 2.4.18_mvl30-ixdp425 #1049 Tue Oct 26 09:34:15 CST 2004 armv5teb
2: sysObjectID.0 (object identifier) enterprises.2021.250.10
3: sysUpTime.0 (timeticks) 0 days 00h:41m:54s.47th (251447)
4: sysContact.0 (octet string) Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
5: sysName.0 (octet string) Moxa
6: sysLocation.0 (octet string) Unknown (configure /etc/snmp/snmp.local.conf)
7: system.8.0 (timeticks) 0 days 00h:00m:00s.22th (22)
8: system.9.1.2.1 (object identifier) mib-2.31
9: system.9.1.2.2 (object identifier) internet.6.3.1
10: system.9.1.2.3 (object identifier) mib-2.49
11: system.9.1.2.4 (object identifier) ip
12: system.9.1.2.5 (object identifier) mib-2.50
13: system.9.1.2.6 (object identifier) internet.6.3.16.2.2.1
14: system.9.1.2.7 (object identifier) internet.6.3.10.3.1.1
15: system.9.1.2.8 (object identifier) internet.6.3.11.3.1.1
16: system.9.1.2.9 (object identifier) internet.6.3.15.2.1.1
17: system.9.1.3.1 (octet string) The MIB module to describe generic objects for network interface sub-layers
18: system.9.1.3.2 (octet string) The MIB module for SNMPv2 entities
19: system.9.1.3.3 (octet string) The MIB module for managing TCP implementations
20: system.9.1.3.4 (octet string) The MIB module for managing IP and ICMP implementations
21: system.9.1.3.5 (octet string) The MIB module for managing UDP implementations
```

22: system.9.1.3.6 (octet string) View-based Access Control Model for SNMP.
23: system.9.1.3.7 (octet string) The SNMP Management Architecture MIB.
24: system.9.1.3.8 (octet string) The MIB for Message Processing and Dispatching.
25: system.9.1.3.9 (octet string) The management information definitions for the SNMP User-based Security Model.
26: system.9.1.4.1 (timeticks) 0 days 00h:00m:00s.04th (4)
27: system.9.1.4.2 (timeticks) 0 days 00h:00m:00s.09th (9)
28: system.9.1.4.3 (timeticks) 0 days 00h:00m:00s.09th (9)
29: system.9.1.4.4 (timeticks) 0 days 00h:00m:00s.09th (9)
30: system.9.1.4.5 (timeticks) 0 days 00h:00m:00s.09th (9)
31: system.9.1.4.6 (timeticks) 0 days 00h:00m:00s.19th (19)
32: system.9.1.4.7 (timeticks) 0 days 00h:00m:00s.22th (22)
33: system.9.1.4.8 (timeticks) 0 days 00h:00m:00s.22th (22)
34: system.9.1.4.9 (timeticks) 0 days 00h:00m:00s.22th (22)
***** SNMP QUERY FINISHED *****

NOTE Click the following links for more information about MIB II and RS-232 like groups:
http://www.faqs.org/rfcs/rfc1213.html
http://www.faqs.org/rfcs/rfc1317.html

→ The DA-660 does NOT support SNMP trap.

OpenVPN

OpenVPN provides two types of tunnels for users to implement VPNS: **Routed IP Tunnels** and **Bridged Ethernet Tunnels**. To begin with, check to make sure that the system has a virtual device /dev/net/tun. If not, issue the following command:

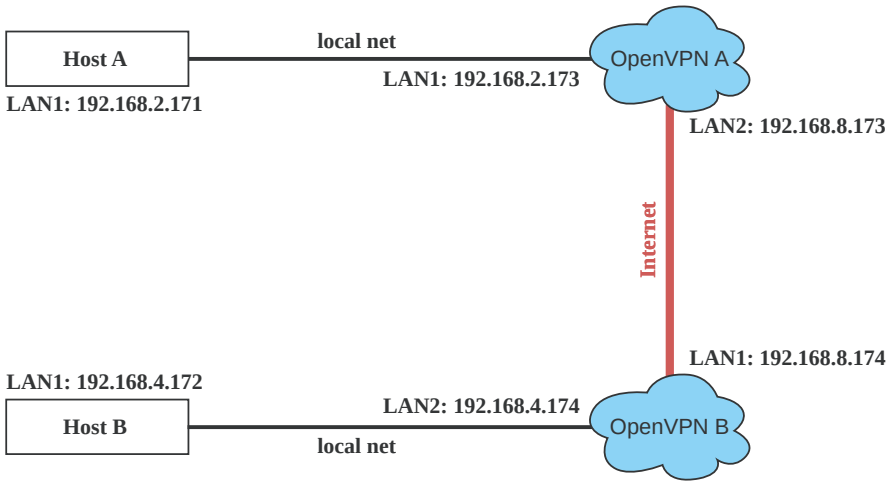
```
# mknod /dev/net/tun c 10 200
```

An Ethernet bridge is used to connect different Ethernet networks together. The Ethernets are bundled into one bigger, "logical" Ethernet. Each Ethernet corresponds to one physical interface (or port) that is connected to the bridge.

On each OpenVPN machine, you should generate a working directory, such as **/etc/openvpn**, where script files and key files reside. Once established, all operations will be performed in that directory.

Setup 1: Ethernet Bridging for Private Networks on Different Subnets

1. Set up four machines, as shown in the following diagram.



Host A (B) represents one of the machines that belongs to OpenVPN A (B). The two remote subnets are configured for a different range of IP addresses. When this setup is moved to a public network, the external interfaces of the OpenVPN machines should be configured for static IPs, or connect to another device (such as a firewall or DSL box) first.

```
# openvpn --genkey --secret secrouter.key
```

Copy the file that is generated to the OpenVPN machine.

2. The **openvpn-bridge** script file located at "/etc/openvpn/" reconfigures interface "eth1" as IP-less, creates logical bridge(s) and TAP interfaces, loads modules, enables IP forwarding, etc.

```
#--------------------------------Start----------------------------

#!/bin/sh

iface=eth1 # defines the internal interface
maxtap=`expr 1` # defines the number of tap devices. I.e., # of tunnels

IPADDR=
NETMASK=
BROADCAST=

# it is not a great idea but this system doesn't support
# /etc/sysconfig/network-scripts/ifcfg-eth1
ifcfg_vpn()
{
while read f1 f2 f3 f4 r3
do
  if [ "$f1" = "iface" -a "$f2" = "$iface" -a "$f3" = "inet" -a "$f4" = "static" ];then
    i=`expr 0`
    while :
    do
      if [ $i -gt 5 ]; then
        break
      fi
      i=`expr $i + 1`
      read f1 f2
      case "$f1" in
        address ) IPADDR=$f2
          ;;
        netmask ) NETMASK=$f2
          ;;
        broadcast ) BROADCAST=$f2
          ;;
      esac
    done
        break
  fi
done < /etc/network/interfaces
}

# get the ip address of the specified interface
mname=
module_up()
{
  oIFS=$IFS
  IFS='
  '
  FOUND="no"
  for LINE in `lsmod`
  do
    TOK=`echo $LINE | cut -d' ' -f1`
    if [ "$TOK" = "$mname" ]; then
      FOUND="yes";
      break;
```

```
    fi
  done
  IFS=$oIFS

  if [ "$FOUND" = "no" ]; then
    modprobe $mname
  fi
}

start()
{
ifcfg_vpn
if [ ! \( -d "/dev/net" \) ]; then
  mkdir /dev/net
fi

if [ ! \( -r "/dev/net/tun" \) ]; then
  # create a device file if there is none
  mknod /dev/net/tun c 10 200
fi

# load modules "tun" and "bridge"
mname=tun
module_up
mname=bridge
module_up
# create an ethernet bridge to connect tap devices, internal interface
brctl addbr br0
brctl addif br0 $iface
# the bridge receives data from any port and forwards it to other ports.

i=`expr 0`
while :
do
  # generate a tap0 interface on tun
  openvpn --mktun --dev tap${i}

  # connect tap device to the bridge
  brctl addif br0 tap${i}

  # null ip address of tap device
  ifconfig tap${i} 0.0.0.0 promisc up

  i=`expr $i + 1`
  if [ $i -ge $maxtap ]; then
    break
  fi
done

# null ip address of internal interface
ifconfig $iface 0.0.0.0 promisc up

# enable bridge ip
ifconfig br0 $IPADDR netmask $NETMASK broadcast $BROADCAST

ipf=/proc/sys/net/ipv4/ip_forward
# enable IP forwarding
echo 1 > $ipf
echo "ip forwarding enabled to"
cat $ipf
}

stop() {
echo "shutdown openvpn bridge."
ifcfg_vpn
i=`expr 0`
while :
do
```

```
  # disconnect tap device from the bridge
  brctl delif br0 tap${i}
  openvpn --rmtun --dev tap${i}

  i=`expr $i + 1`
  if [ $i -ge $maxtap ]; then
    break
  fi
done
brctl delif br0 $iface
brctl delbr br0
ifconfig br0 down
ifconfig $iface $IPADDR netmask $NETMASK broadcast $BROADCAST
killall -TERM openvpn
}

case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    stop
    start
    ;;
  *)
    echo "Usage: $0 [start|stop|restart]"
    exit 1
esac
exit 0
#--------------------------------- end ----------------------------
```

Create link symbols to enable this script at boot time:

```
# ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc3.d/S32vpn-br # for example
# ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc6.d/K32vpn-br # for example
```

3. On machine OpenVPN A, modify the remote address in the configuration file, **/etc/openvpn/tap0-br.conf**.

```
# /etc/openvpn/tap0-br.conf
# point to the peer
remote 192.168.8.174
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
```

Then modify the routing table in **/etc/openvpn/tap0-br.sh** script file.

```
#---------------------------------Start-----------------------------
#!/bin/sh
# /etc/openvpn/tap0-br.sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 dev br0
#--------------------------------- end ----------------------------
```

On machine OpenVPN B, modify the remote address in the configuration file, **/etc/openvpn/tap0-br.conf**.

```
# /etc/openvpn/tap0-br.conf
# point to the peer
remote 192.168.8.173
```

```
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5 tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
```

Then modify the routing table in **/etc/openvpn/tap0-br.sh** script file.

```
#---------------------------------Start-----------------------------
#!/bin/sh
# /etc/openvpn/tap0-br.sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 dev br0
#--------------------------------- end ----------------------------
```

NOTE Select cipher and authentication algorithms by specifying "cipher" and "auth". To see with algorithms are available, type:

```
# openvpn --show-ciphers
# openvpn --show—auths
```

4. After configuring the remote peer, we can load the bridge into kernel, reconfigure eth1 and enable IP forwarding on both **OpenVPN** machine.

   ```
   # /etc/openvpn/openvpn-bridge start
   ```

   Then start both of OpenVPN peers,

   ```
   # openvpn --config /etc/openvpn/tap0-br.conf &
   ```

   If you see the line "Peer Connection Initiated with 192.168.8.173:5000" on each machine, the connection between OpenVPN machines has been established successfully on UDP port 5000.

NOTE You can create link symbols to enable the **/etc/openvpn/openvpn-bridge** script at boot time:

```
# ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc3.d/S32vpn-br
# ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc6.d/K32vpn-br
```

5. On each OpenVPN machine, check the routing table by typing the command:

   ```
   # route
   ```

Destination	Gateway	Genmsk	Flags	Metric	Ref	Use	Iface
192.168.4.0	*	255.255.255.0	U	0	0	0	br0
192.168.2.0	*	255.255.255.0	U	0	0	0	br0
192.168.8.0	*	255.255.255.0	U	0	0	0	eth0

   Interface **eth1** is connected to the bridging interface **br0**, to which device **tap0** also connects, whereas the virtual device **tun** sits on top of **tap0**. This ensures that all traffic from internal networks connected to interface **eth1** that come to this bridge write to the TAP/TUN device that the OpenVPN program monitors. Once the OpenVPN program detects traffic on the virtual device, it sends the traffic to its peer.

6. To create an indirect connection to Host B from Host A, you need to add the following routing item:

   ```
   route add –net 192.168.4.0 netmask 255.255.255.0 dev eth0
   ```

   To create an indirect connection to Host A from Host B, you need to add the following routing item:

   ```
   route add –net 192.168.2.0 netmask 255.255.255.0 dev eth0
   ```

   Now ping Host B from Host A by typing:

```
ping 192.168.4.174
```

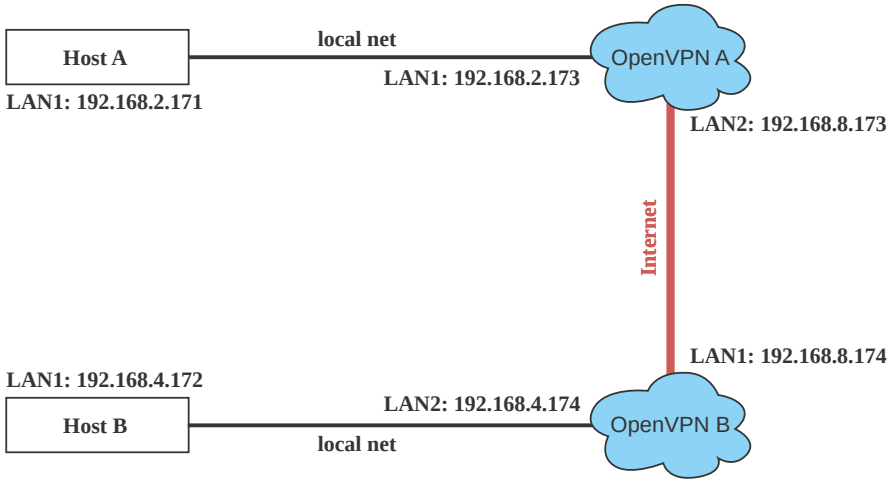
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

7. To shut down OpenVPN programs, type the command:

```
# /etc/openvpn/openvpn-bridge stop
```

Setup 2: Ethernet Bridging for Private Networks on the Same Subnet

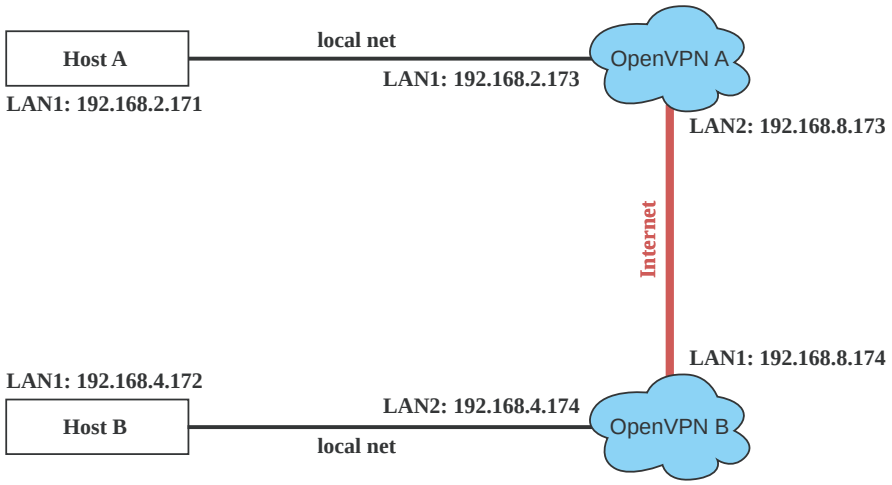
1. Set up four machines as shown in the following diagram:



2. The configuration procedure is almost the same as for the previous example. The only difference is that you must comment out the parameter **up** in **/etc/openvpn/tap0-br.conf** and **/etc/openvpn/tap0-br.conf**.

Setup 3: Routed IP

1. Set up four machines as shown in the following diagram:



2. Create a configuration file named "A-tun.conf" and an executable script file named "A-tun.sh".

```
# point to the peer
remote 192.168.8.174
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.2.173 192.168.4.174
up /etc/openvpn/A-tun.sh
```

Then modify the routing table in **/etc/openvpn/tun.sh** script file.

```
#---------------------------------Start-----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 gw $5
#--------------------------------- end -----------------------------
```

Create a configuration file named **B-tun.conf** and an executable script file named **B-tun.sh** on OpenVPN B:

```
remote 192.168.8.173
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.4.174 192.168.2.173
up /etc/openvpn/B-tun.sh
```

```
#---------------------------------Start-----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 gw $5
#--------------------------------- end -----------------------------
```

NOTE The command **ifconfig** defines the first argument as the local internal interface and the second argument as the internal interface at the remote peer.

NOTE **$5** is the argument that the OpenVPN program passes to the script file. Its value is the second argument of ifconfig in the configuration file.

3. Check the routing table after you run the OpenVPN programs, by typing the command:

```
# route
```

Destination	Gateway	Genmsk	Flags	Metric	Ref	Use	Iface
192.168.4.174	*	255.255.255.255	UH	0	0	0	tun0
192.168.4.0	192.168.4.174	255.255.255.0	UG	0	0	0	tun0
192.168.2.0	*	255.255.255.0	U	0	0	0	eth1
192.168.8.0	*	255.255.255.0	U	0	0	0	eth0

5

Programmer's Guide

This chapter includes important information for programmers.

The following functions are covered in this chapter:

- **Flash Memory Map**
- **Linux Tool Chain Introduction**
- **Debugging with GDB**
- **Device API**
- **RTC (Real Time Clock)**
- **Buzzer**
- **WDT (Watch Dog** Timer)
- **UART**
- **LCM**
- **KeyPad**
- **Makefile Example**

Flash Memory Map

Partition sizes are hard coded into the kernel binary. To change the partition sizes, you will need to rebuild the kernel. The flash memory map is shown in the following table.

Address	Size	Contents
0x00000000 – 0x0005FFFF	384 KB	Boot Loader—Read ONLY
0x00060000 – 0x0015FFFF	1 MB	Kernel object code—Read ONLY
0x00160000 – 0x0055FFFF	4 MB	Mini root file system (EXT2)—Read ONLY
0x00560000 – 0x01F5FFFF	26 MB	User root file system (JFFS2)—Read/Write
0x01F60000 – 0x01FBFFFF	384 KB	Not used
0x01FC0000 – 0x01FDFFFF	128 KB	Boot Loader configuration—Read ONLY
0x01FE0000 – 0x01FFFFFF	128 KB	Boot Loader directory—Read ONLY

Mount the user file system to /**mnt/usrdisk** with the root file system. Check to see if the user file system was mounted correctly. If the user file system is okay, the kernel will change the root file system to **/mnt/usrdisk**. If the user file system is not okay, the kernel will use the default Moxa file system. To finish the boot process, run the **init** program.

NOTE

1. The default Moxa file system only enables the network and CF. It lets users recover the user file system when it fails.
2. The user file system is a complete file system. Users can create and delete directories and files (including source code and executable files) as needed.
3. Users can create the user file system on the PC host or target platform, and then copy it to the DA-660.

Linux Tool Chain Introduction

To ensure that an application will be able to run correctly when installed on the DA-660, you must ensure that it is compiled and linked to the same libraries that are on the DA-660. This is particularly true when the RISC Xscale processor architecture of the DA-660 differs from the CISC x86 processor architecture of the host system, but it is also true if the processor architecture is the same.

The host tool chain that comes with the DA-660 contains a suite of cross compilers and other tools, as well as the libraries and headers that are necessary to compile applications for the DA-660. The host environment must be running Linux to install the DA-660 GNU Tool Chain. We have confirmed that the following Linux distributions can be used to install the tool chain:

Redhat 7.3/8.0/9.0, Fedora core 1 & 2.

The Tool Chain will need about 100 MB of hard disk space on your PC. The DA-660 Tool Chain is located on the DA-660 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#rpm –ivh /mnt/cdrom/mxscaleb-3.3.2-x.i386.rpm
```

Wait for a few minutes while the Tool Chain is installed automatically on your Linux PC. Once the host environment has been installed, add the directory **/usr/local/mxscaleb/bin** to your path and the directory **/usr/local/mxscaleb/man** to your manual path. You can do this temporarily for the current login session by issuing the following commands:

```
#export PATH="/usr/local/mxscaleb/bin:$PATH"
#export MANPATH="/usr/local/mxscaleb/man:$PATH"
```

Alternatively, you can add the same commands to **$HOME/.bash_profile** to cause it to take effect for all login sessions initiated by this user.

Obtaining help

Use the Linux man utility to obtain help on many of the utilities provided by the tool chain. For example to get help on the armv5b-linux-gcc compiler, issue the command:

#man armv5b-linux-gcc

Cross Compiling Applications and Libraries

To compile a simple C application, just use the cross compiler instead of the regular compiler:

```
#mxscaleb-gcc  –o example –Wall –g –O2 example.c
#mxscaleb-strip  –s example
#mxscaleb-gcc  -ggdb –o example-debug example.c
```

Tools Available in the Host Environment

Most of the cross compiler tools are the same as their native compiler counterparts, but with an additional prefix that specifies the target system. In the case of x86 environments, the prefix is i386-linux- and in the case of the DA-660 Xscale boards, it is **mxscaleb-**.

For example the native C compiler is gcc and the cross C compiler for Xscale in DA-660 is **mxscaleb-gcc**.

The following cross compiler tools are provided:

ar	Manage archives (static libraries)
as	Assembler
c++, g++	C++ compiler
cpp	C preprocessor
gcc	C compiler
gdb	Debugger
ld	Linker
nm	Lists symbols from object files
objcopy	Copies and translates object files
objdump	Displays information about object files
ranlib	Generates indexes to archives (static libraries)
readelf	Displays information about ELF files
size	Lists object file section sizes
strings	Prints strings of printable characters from files (usually object files)
strip	Removes symbols and sections from object files (usually debugging information)

Debugging with GDB

First compile the program must with option -ggdb. Use the following steps:

1. To debug a program called **hello-debug** on the target, use the command:

   ```
   #gdbserver 192.168.4.142:2000 hello-debug
   ```

   This is where 2000 is the network port number on which the server waits for a connection from the client. This can be any available port number on the target. Following this is the name of the program to be debugged (hello-debug), plus that program's arguments. Output similar to the following will be sent to the console:

   ```
   Process hello-debug created; pid=38
   ```

2. Use the following command on the host to change to the directory that contains **hello-debug**:

   ```
   cd /my_work_directory/myfilesystem/testprograms
   ```

3. Enter the following command:

   ```
   #ddd --debugger mxscaleb-gdb hello-debug &
   ```

4. Enter the following command at the GDB, DDD command prompt:

   ```
   Target remote 192.168.4.99:2000
   ```

   The command produces another line of output on the target console, similar to the following:

   ```
   Remote debugging using 192.168.4.99:2000
   ```

   192.168.4.99 is the machine's IP address, and 2000 is the port number. You can now begin debugging in the host environment using the interface provided by DDD.

5. Set a breakpoint on main by double clicking or entering b main on the command line.

6. Click the **cont** button

Device API

The DA-660 supports control devices with the **ioctl** system API. You will need to use **include <moxadevice.h>**, and use the following **ioctl** function.

```
int ioctl(int d, int request,…);
  Input: int d - open device node return file handle
    int request – argument in or out
```

Use the ioctl man page for detailed documentation:

```
#man ioctl
```

RTC (Real Time Clock)

The device node is located at **/dev/rtc**. The DA-660 supports Linux standard simple RTC control. You must include **<linux/rtc.h>**.

1. Function: RTC_RD_TIME

   ```
   int ioctl(fd, RTC_RD_TIME, struct rtc_time *time);
   ```

   Description: read time information from RTC. This returns the value on argument 3.

2. Function: RTC_SET_TIME

   ```
   int ioctl(fd, RTC_SET_TIME, struct rtc_time *time);
   ```

   Description: set RTC time. Argument 3 will be passed to the RTC.

Buzzer

The device node is located at **/dev/console**. The DA-660 supports Linux standard buzzer control, with its buzzer running at a fixed frequency of 100 Hz. You must use **include <sys/kd.h>**.

Function: KDMKTONE

```
ioctl(fd, KDMKTONE, unsigned int arg);
```

Description: The buzzer's behavior is determined by the argument arg. The **high word** part of arg indicates the length of time the buzzer will sound, and the **low word** part indicates the frequency.

The buzzer's on/off behavior is controlled by software. If you call the **ioctl** function, you MUST set the frequency at 100 Hz. If you use a different frequency, the system could crash.

WDT (Watch Dog Timer)

1. **Introduction**

   The WDT works like a watch dog function. You can enable it or disable it. If the user enables WDT but the application does not acknowledge it, then the system will reboot. You can set the acknowledgement time from a minimum of 50 msec to a maximum of 60 seconds.

2. **How the WDT works**

   The sWatch Dog is enabled when the system boots up. The kernel will auto acknowledge it. The user application must also acknowledge it. When the user application does not acknowledge it, the system will reboot.

   Kernel boot
   .....
   ....

   User application running and enable user acknowledgement

   ....
   ....

3. **The user API**

   The user application must include **<moxadevic.h>**, and **link moxalib.a**. A Makefile example is shown below:

```
all:
  mxscaleb-gcc –o xxxx  xxxx.c -lmoxalib
```

`int swtd_open(void)`

Description

Open the file handle to control the sWatch Dog. Keep the file handle to do other operations.

Input

None

Output

The return value is the file handle. In case of an error, it will return a value < 0.

You can get the error from **errno()**.

`int swtd_enable(int fd, unsigned long time)`

Description

Enables the Watch Dog for the application. The application must acknowledge this, else the system will reboot.

Input

int fd - the file handle, from the swtd_open() return value.

unsigned long time - The time you wish to acknowledge the Watch Dog periodically. You must acknowledge the Watch Dog before timeout. If you do not, the system will be rebooted automatically. The minimal time is 50 msec, the maximum time is 60 seconds. The time unit is msec.

Output

The output will be zero if everything is OK. Any other value indicates an error. Use the **errno()** function to get the actual error description.

`int swtd_disable(int fd)`

Description

Disable the application Watch Dog. The kernel will auto acknowledge the action.

Input

int fd - the file handle from swtd_open() return value.

Output

The output will be zero if everything is OK. Any other value indicates an error. Use the **errno()** function to get the actual error description.

`int swtd_get(int fd, int *mode, unsigned long *time)`

Description

Get the current settings.

mode –

1 if user application enables the Watch Dog: need to acknowledge the action.
0 if user application disables Watch Dog: does not need to acknowledge the action.

time – The time period to acknowledge the Watch Dog.

Input

int fd - the file handle from swtd_open() return value.

int *mode - the function will be return the status - enable or disable the user application; indicate whether an acknowledgement is necessary.

unsigned long *time - the function will return the current time period.

Output

The output will be zero if everything is OK. Any other value indicates an error. Use the errno() function to get the actual error description.

`int swtd_ack(int fd)`

Description

Acknowledge the Watch Dog. When the user application enables the Watch Dog, it needs to call this function periodically with the user predefined time in the application program.

Input

int fd - the file handle from swtd_open() return value.

Output

The output will be zero if everything is OK. Any other value indicates an error. Use the errno() function to get the actual error description.

`int swtd_close(int fd)`

Description

Close the file handle.

Input

int fd - the file handle from **swtd_open()** return value.

Output

The output will be zero if everything is OK. Any other value indicates an error. Use the errno() function to get the actual error description.

4. Special Note

When you kill the application with -9 or kill without any option or kill with a **Ctrl+c**, the kernel will auto acknowledge the Watch Dog.

When your application enables the Watch Dog and does not acknowledge it, your application may have a logical error, or your application has made a core dump. The kernel will not auto acknowledge this. This can cause a serious problem, causing your system to reboot again and again.

5. User application example

Example 1:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <moxadevice.h>

int main(int argc, char *argv[])
{
  int fd;

  fd = swtd_open();
  if ( fd < 0 ) {
    printf("Open sWatchDog device fail !\n");
    exit(1);
  }
  swtd_enable(fd, 5000); // enable it and set it 5 seconds
  while ( 1 ) {
    // do user application want to do
    …..
    ….
    swtd_ack(fd);
    …..
    ….
  }
  swtd_close(fd);
  exit(0);

}
```

The makefile is shown below:

```
all:
  mxscaleb-gcc –o xxxx xxxx.c –lmoxalib
```

Example 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <moxadevice.h>

static void mydelay(unsigned long msec)
{
  struct timeval time;

  time.tv_sec = msec / 1000;
  time.tv_usec = (msec % 1000) * 1000;
  select(1, NULL, NULL, NULL, &time);
}

static int swtdfd;
static int stopflag=0;

static void stop_swatchdog()
{
  stopflag = 1;
}

static void do_swatchdog(void)
{
  swtd_enable(swtdfd, 500);
  while ( stopflag == 0 ) {
    mydelay(250);
    swtd_ack(swtdfd);
  }
  swtd_disable(swtdfd);
}

int main(int argc, char *argv[])
{
  pid_t sonpid;

  signal(SIGUSR1, stop_swatchdog);
  swtdfd = swtd_open();
  if ( swtdfd < 0 ) {
    printf("Open sWatchDog device fail !\n");
    exit(1);
  }
  if ( (sonpid=fork()) == 0 )
    do_swatchdog();
  // do user application main function
  …..
  …..
  …..
  // end user application
  kill(sonpid, SIGUSR1);
  swtd_close(swtdfd);
  exit(1);
}
```

The Makefile is shown below:

```
all:
  mxscaleb-gcc –o xxxx xxxx.c –lmoxalib
```

UART

The normal tty device node is located at **/dev/ttyM0 … ttyM7,** and the modem tty device node is located at **/dev/cum0 … cum7**.

The DA-660 supports Linux standard termios control. The MOXA UART Device API allows you to configure ttyM0 to ttyM7 as RS-232, RS-422, 4-wire RS-485, or 2-wire RS-485. The DA-660 supports RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

You must use **include <moxadevice.h>**.

```
#define RS232_MODE     0
#define RS485_2WIRE_MODE    1
#define RS422_MODE     2
#define RS485_4WIRE_MODE    3
```

1. Function: MOXA_SET_OP_MODE
   `int ioctl(fd, MOXA_SET_OP_MODE, &mode)`

   Description

   Sets the interface mode. The mode in parameter 3 will be passed to the UART device driver which will change to it.

2. Function: MOXA_GET_OP_MODE
   `int ioctl(fd, MOXA_GET_OP_MODE, &mode)`

   Description

   Gets the interface mode. The mode will be returned in parameter 3.

There are two Moxa private ioctl commands for setting up special baudrates.

Function: MOXA_SET_SPECIAL_BAUD_RATE
Function: MOXA_GET_SPECIAL_BAUD_RATE

If you use this ioctl to set a special baudrate, the termios cflag will be B4000000, in which case the B4000000 defined will be different. If the baudrate you get from termios (or from calling tcgetattr()) is B4000000, you must call ioctl with MOXA_GET_SPECIAL_BAUD_RATE to get the actual baudrate.

Example to set the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
term. c_cflag &= ~(CBAUD | CBAUDEX);
term.c_cflag |= B4000000;
tcsetattr(fd, TCSANOW, &term);
speed = 500000;
ioctl(fd, MOXA_SET_SPECIAL_BAUD_RATE, &speed);
```

Example to get the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
if ( (term.c_cflag & (CBAUD|CBAUDEX)) != B4000000 ) {
  // follow the standard termios baudrate define
} else {
  ioctl(fd, MOXA_GET_SPECIAL_BAUD_RATE, &speed);
}
```

Baudrate inaccuracy

Divisor = 921600/Target Baudrate. (Only Integer part)
ENUM = 8 * (921600/Targer - Divisor) (Round up or down)
Inaccuracy = (Target Baudrate – 921600/(Divisor + (ENUM/8))) * 100%
E.g.,
To calculate 500000 bps
Divisor = 1, ENUM = 7,
Inaccuracy = 1.7%
*The Inaccuracy should be less than 2% for reliability.

Special Note

1. If the target baudrate is not a special baudrate (e.g. 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600), the termios cflag will be set to the same flag.
2. If you use stty to get the serial information, you will get a speed equal to 0.

LCM

The DA-660 only supports text mode display, with screen size of 16 cols by 8 rows. The device node is **/dev/lcm**. See the examples given below. We provide a private struct defined as follows:

```
typedef struct lcm_xy {
   int x; // col value, the arrange is 0 – 15
   int y; // raw value, the arrange is 0 – 7
} lcm_xy_t;
```

Examples

```
int ioctl(fd, IOCTL_LCM_GOTO_XY, lcm_xy_t *pos);
```

Move the cursor position to x(col),y(raw) position. The argument 3 is the new position value.

```
int ioctl(fd, IOCTL_LCM_CLS, NULL);
```

Clears the LCM display.

```
int ioctl(fd, IOCTL_LCM_CLEAN_LINE, NULL);
```

Change one line to all spaces in the current row, and move the cursor to the 0 column of this row.

```
int ioctl(fd, IOCTL_LCM_GET_XY, lcm_xy_t *pos);
```

Get the current cursor position. The value will be returned in argument 3.

```
int ioctl(fd, IOCTL_LCM_BACK_LIGH_ON, NULL);
```

Turns the LCM backlight on.

```
int ioctl(fd, IOCTL_LCM_BACK_LIGHT_OFF, NULL);
```

Turns the LCM backlight off.

KeyPad

The device node is **/dev/keypad**. The key value is defined in moxadevice.h.

```
int ioctl(fd, IOCTL_KEYPAD_HAS_PRESS, int *flag);
```

Checks how many keys have been pressed. Argument 3 returns the number of pressed keys. 0 means no keys were pressed.

```
int ioctl(fd, IOCTL_KEYPAD_GET_KEY, int *key);
```

Gets the value of the last key that was pressed. This function only reads one key value for each

function call. The value of the key value is returned in argument 3.

Special Note

1. The DA-660's kernel will store the "pressed key history" in a buffer. The maximum buffer size is 31 keys. If the buffer overflows, the first key of the 31 keys that was pressed will be dropped, without sounding the buzzer.
2. Currently, the DA-660 does NOT support pressing more than 1 key at the same time.

Makefile Example

The following Makefile file example codes are copied from the Hello example on the DA-660's CD-ROM.

```
CC = /usr/local/mxscaleb/mxscaleb-gcc
CPP = /usr/local/mxscaleb/mxscaleb-gcc
SOURCES = hello.c

OBJS =  $(SOURCES:.c=.o)

all: hello

hello: $(OBJS)
$(CC) -o $@ $^ $(LDFLAGS) $(LIBS)

clean:
rm -f $(OBJS) hello core *.gdb
```

6

UC Finder

UC Finder comes in handy if you forget the IP address of the target computer when you have a demand for troubleshooting field problems. This utility works by sending a broadcast message over the LAN to search for IP addresses of target computers. Two versions of UC Finder are provided. The GUI version works in Windows environments, and the command line utility works in Linux environments.

The following topics are covered in this chapter:

- **Windows UC Finder**
  - Installing the Software
  - Broadcast Search
- **Linux UC Finder**

Windows UC Finder

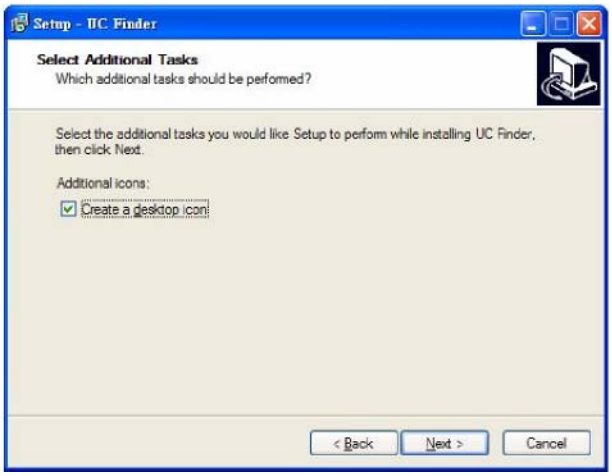
The UC Finder utility is used to search the LAN or intranet for Moxa embedded computers.

Installing the Software

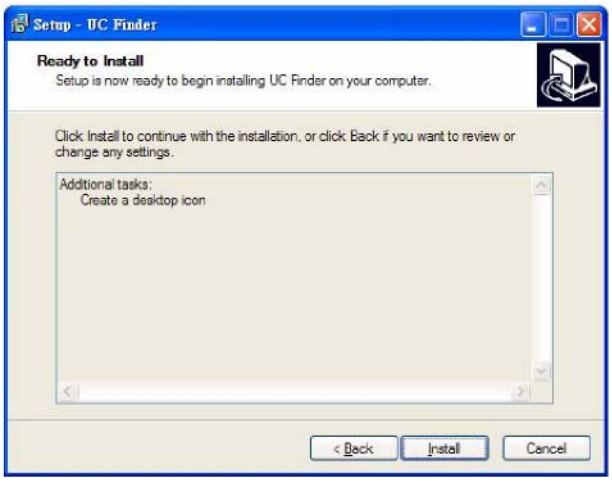
1. Once the Setup program starts running, click **Next** to proceed.

Setup - UC Finder
Welcome to the UC Finder Setup Wizard
This will install UC Finder on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

2. Click **Next** when the **Select Additional Tools** window opens to proceed with the installation.


Setup - UC Finder
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing UC Finder, then click Next.
Additional icons:
Create a desktop icon
< Back
Next >
Cancel

3. Click **Next** to install program files in the default directory, or select an alternative location.


Setup - UC Finder
Ready to Install
Setup is now ready to begin installing UC Finder on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Additional tasks:
Create a desktop icon
< Back
Install
Cancel

4. Click **Finish** to complete the installation of UC Finder.

Setup - UC Finder
Completing the UC Finder Setup Wizard
Setup has finished installing UC Finder on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Launch UC Finder
Finish

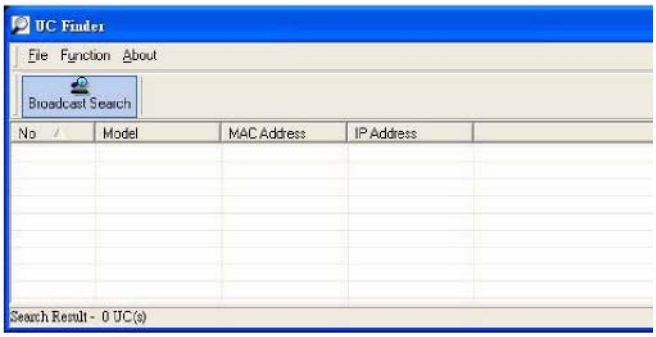
Broadcast Search

The **Broadcast Search** function is used to locate all the DA-660 units that are connected to the same LAN as your Windows computer.

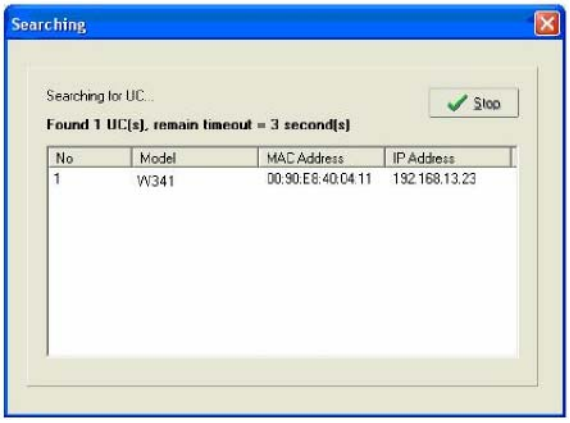
Since the Broadcast Search function searches by MAC address and not IP address, all the

DA-660s connected to the LAN will be found, regardless of whether or not they are on the same subnet as the host.

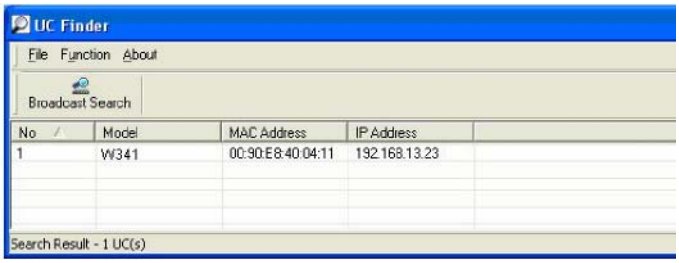
1. Click the **Broadcast Search** button to start searching.



2. The **Searching** window displays the **Model, MAC Address**, and **IP Address** of devices that were located.



3. When the search is complete, the same information is displayed in the UC Finder window.



NOTE UC Finder is designed to determine the IP addresses of all UC units connected to the same LAN as the host that is running UC Finder. To configure UC's IP addresses or other configuration parameters, use Telnet over the network, or connect directly to the serial Console port to access the DA-660's Console utility.

Linux UC Finder

Copy **ucfinder** from the Document and Software CD to your Linux PC, and then use the following command to start the program. UC Finder will automatically locate all the DA-660 units on the LAN, and then report their IP addresses.

```
#./ucfinder
```

A

System Commands

busybox (V0.60.4): Linux normal command utility collection

File Manager

1.	**cp**	**copy file**
2.	**ls**	**list file**
3.	**ln**	**make symbolic link file**
4.	**mount**	**mount and check file system**
5.	**rm**	**delete file**
6.	**chmod**	**change file owner, group, and user**
7.	**chown**	**change file owner**
8.	**chgrp**	**change file group**
9.	**sync**	**sync file system, let system file buffer be saved to hardware**
10.	**mv**	**move file**
11.	**pwd**	**displays the current working directly**
12.	**df**	**displays the amount of free space on the device**
13.	**mkdir**	**make new directory**
14.	**rmdir**	**delete directory**

Editor

1.	**vi**	**text editor**
2.	**cat**	**dump file context**
3.	**zcat**	**compress or expand files**
4.	**grep**	**search file for a specific pattern**
5.	**cut**	**get string on file**
6.	**find**	**find files**
7.	**more**	**dump file page by page**
8.	**test**	**test if file exists or not**
9.	**sleep**	**sleep (seconds)**
10.	**echo**	**echo string**

Network

1.	**ping**	**ping to test network**
2.	**route**	**routing table manager**
3.	**netstat**	**display network status**
4.	**ifconfig**	**set network IP address**
5.	**traceroute**	**trace route**
6.	**tftp**	**trivial file transfer protocol**
7.	**telnet**	**A terminal emulation program for TCP/IP network**
8.	**ftp**	**file transfer protocol**

Process

1.	**kill**	**kill process**
2.	**ps**	**display now running process**

Other

1.	**dmesg**	**dump kernel log message**
2.	**sty**	**to set serial port**
3.	**zcat**	**dump .gz file context**
4.	**mknod**	**make device node**
5.	**free**	**display system memory usage**
6.	**date**	**print or set the system date and time**
7.	**env**	**run a program in a modified environment**
8.	**clear**	**clear the terminal screen**
9.	**reboot**	**reboot / power off/on the server**
10.	**halt**	**halt the server**
11.	**du**	**estimate file space usage**
12.	**gzip, gunzip**	**compress or expand files**
13.	**hostname**	**show system's host name**

Moxa Special Utilities

1.	**backupfs**	**backup file system (user directory)**
2.	**bf**	**built the file system (user directory)**
3.	**kversion**	**show kernel version**
4.	**cat /etc/version**	**show user directory version**
5.	**upramdisk**	**mount ramdisk**
6.	**downramdisk**	**unmount ramdisk**

B

Windows Tool Chain Introduction

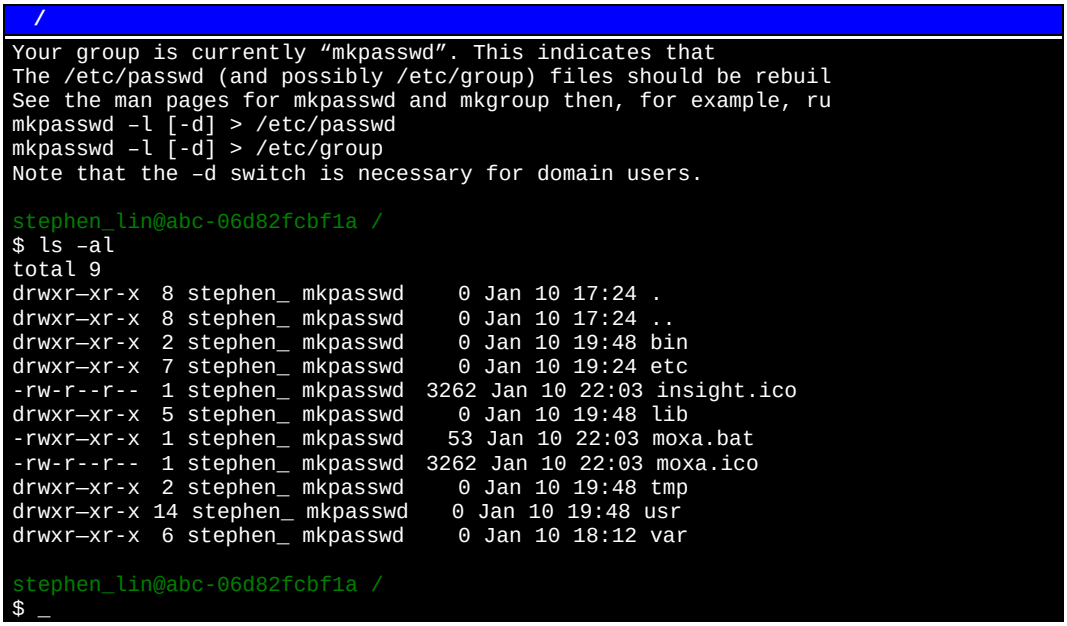
The DA-660's Windows Tool Chain is a cross platform development environment that simulates the Linux root file system, allowing users to develop applications on a Windows PC.

The following topics are covered in this appendix:

- **Introduction**
- **Installation Procedure**
- **Using the BASH Shell**
- **Compilation**
- **GDB Debug Tool—Insight**

Introduction

The DA-660's Windows Tool Chain is a cross platform development environment that simulates the Linux root file system, allowing users to develop applications in a Windows PC environment. Install the Windows tool chain to set up a Linux-like environment on your Windows machine. The following figure shows an example of what the Windows Tool Chain environment looks like.


/
Your group is currently "mkpasswd". This indicates that
The /etc/passwd (and possibly /etc/group) files should be rebuil
See the man pages for mkpasswd and mkgroup then, for example, ru
mkpasswd –l [-d] > /etc/passwd
mkpasswd –l [-d] > /etc/group
Note that the –d switch is necessary for domain users.
stephen_lin@abc-06d82fcbf1a /
$ ls –al
total 9
drwxr–xr-x 8 stephen_ mkpasswd 0 Jan 10 17:24 .
drwxr–xr-x 8 stephen_ mkpasswd 0 Jan 10 17:24 ..
drwxr–xr-x 2 stephen_ mkpasswd 0 Jan 10 19:48 bin
drwxr–xr-x 7 stephen_ mkpasswd 0 Jan 10 19:24 etc
-rw-r--r-- 1 stephen_ mkpasswd 3262 Jan 10 22:03 insight.ico
drwxr–xr-x 5 stephen_ mkpasswd 0 Jan 10 19:48 lib
-rwxr–xr-x 1 stephen_ mkpasswd 53 Jan 10 22:03 moxa.bat
-rw-r--r-- 1 stephen_ mkpasswd 3262 Jan 10 22:03 moxa.ico
drwxr–xr-x 2 stephen_ mkpasswd 0 Jan 10 19:48 tmp
drwxr–xr-x 14 stephen_ mkpasswd 0 Jan 10 19:48 usr
drwxr–xr-x 6 stephen_ mkpasswd 0 Jan 10 18:12 var
stephen_lin@abc-06d82fcbf1a /
$ _

Your Windows OS must satisfy the following requirements.

1. Operating System: Windows 2000 workstation or Windows XP professional.
2. Minimum of 500 MB of free hard drive space to install the tool chain and develop programs. All of the disk space must be on a single drive.
3. CD-ROM or equivalent.
4. Ethernet connect to download the application program to DA-660.
5. Be able to log in as administrator.
6. Use a Windows username without spaces.

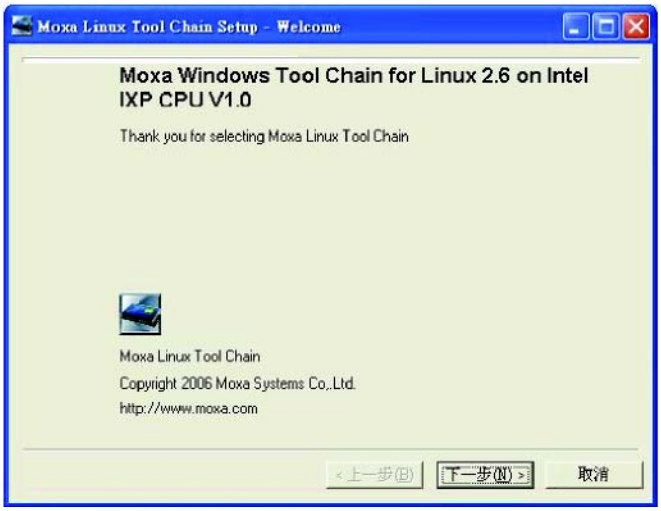
You will be using a BASH shell window to enter commands. In addition, for editing text files, such as configuration files, you should use the **vi** editor. Do NOT use WordPad, which could cause problems when the files are transferred to a true Linux environment.

Installation Procedure

1. Select the Root Directory.

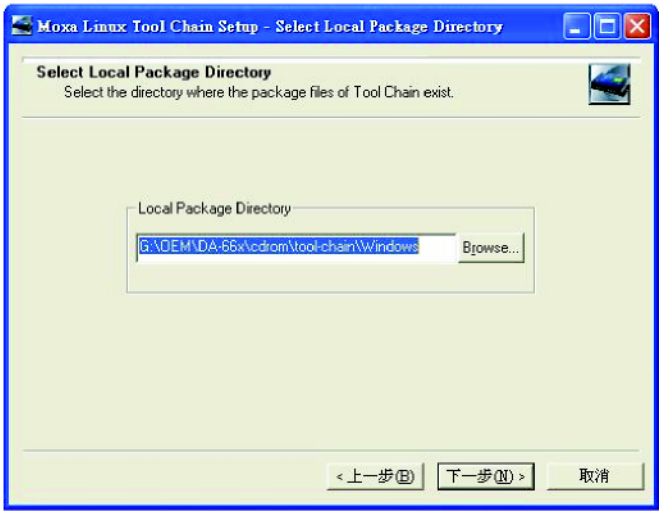
   Note
   If the directory C:\UC was created during a previous installation, it will be displayed as the Root Directory. Use this directory if you do not wish to re-install the UC windows tool chain again.

   Under **Install For, All Users (Recommended)** is selected by default. Under **Default Text File Type, Unix (Recommended)** is selected by default.

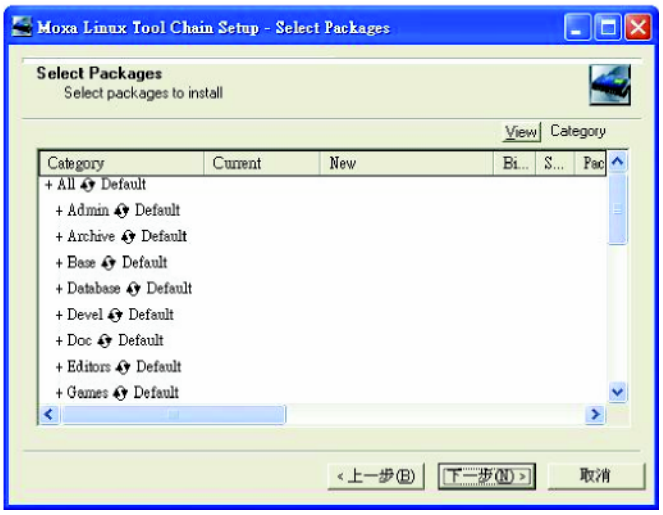

Moxa Linux Tool Chain Setup - Welcome
Moxa Windows Tool Chain for Linux 2.6 on Intel IXP CPU V1.0
Thank you for selecting Moxa Linux Tool Chain
Moxa Linux Tool Chain
Copyright 2006 Moxa Systems Co,.Ltd.
http://www.moxa.com
< 上一步(B)
下一步(N) >
取消

2. By default, the **Local Package Directory** is the same as on the CD-ROM: **\\tool-chain\windows\compiler**. Click on **Next** to proceed.

   Note: It may take anywhere from 5 to 20 minutes to check the packages in the CD-ROM after you click **Next**.

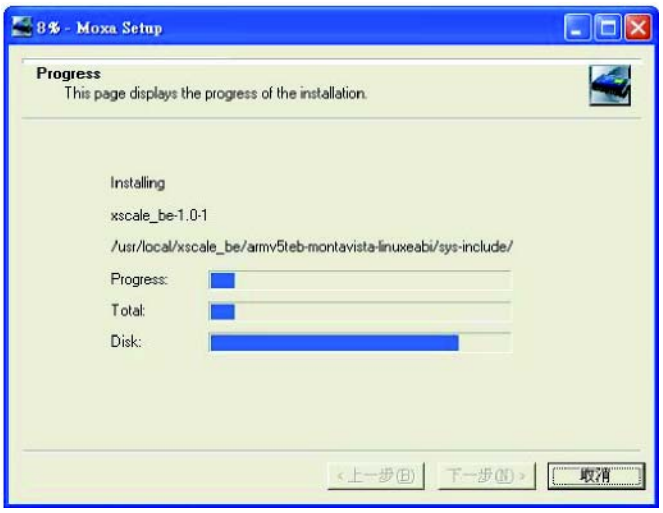

Moxa Linux Tool Chain Setup - Select Local Package Directory
Select Local Package Directory
Select the directory where the package files of Tool Chain exist.
Local Package Directory
G:\OEM\DA-66x\cdrom\tool-chain\Windows
Browse...
< 上一步(B)
下一步(N) >
取消

3. All required package are selected by default. Click **Next** to continue.

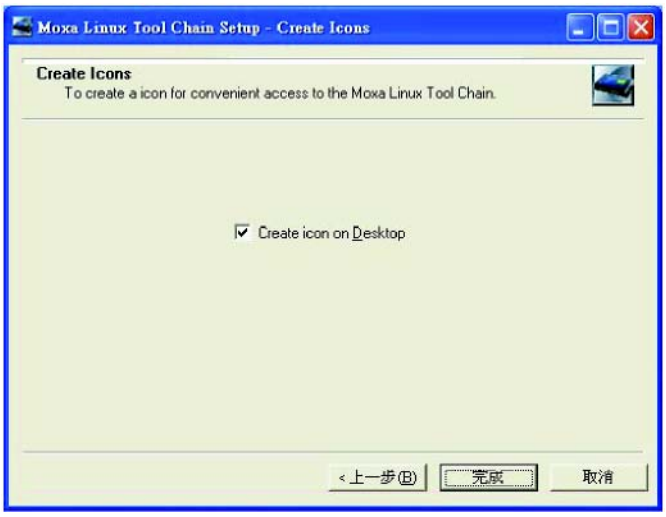


- **Category column:** Shows a hierarchical display of packages with category name highest and package name lowest. Click the plus sign (+) next to a category name to open the category and see the packages within that category.
- **Base Category:** List all packages that will be installed by default (along with any dependencies). If you view the Base column for the Base category, you should see every package selected for installation.
- **Packages:** In the packages listing field, package names are arranged by Category/Full/Partial according to the **View by** button. When viewing by Category, click the plus sign (+) to open or close the packages under that category. This is the same list you see in flat form when viewing by Full/Partial. By default, named packages are installed.
- **Install:** If the package was not previously installed, select this option to install the package now.
- **Reinstall:** If the package was previously installed, select this option to install it again. This will overwrite the previous installation.
- **Uninstall:** If the package was previously installed, select this option if you do NOT want to make any changes.
- **Skip:** Ignores a package entirely, regardless of whether it was previously installed or uninstalled. Packages marked "Skip" are omitted from the Partial display.

4. The **Progress** window will appear. The installer will install all packages that were selected. This process could take from 5 to 30 minutes, depending on the speed of your system. When the installation is complete, the **Complete the Installation** window will appear.



5. Select the **Create icon on Desktop** option to place a **Moxa BASH Shell** icon on your desktop, and then click **Finish**.


Moxa Linux Tool Chain Setup - Create Icons
Create Icons
To create a icon for convenient access to the Moxa Linux Tool Chain.
Create icon on Desktop
< 上一步(B)
完成
取消

6. Click **OK** to close the **Installation Complete** window.

Using the BASH Shell

A command-line interface is used to develop applications with the tool chain. To access the command line, you need to launch a BASH shell window.

To open a BASH shell window, choose **Start Menu → Programs → UC Tool Chain → Moxa Bash Shell**, or click the **Moxa Bash Shell** icon on your desktop.

```
/
Your group is currently "mkpasswd". This indicates that
The /etc/passwd (and possibly /etc/group) files should be rebuil
See the man pages for mkpasswd and mkgroup then, for example, ru
mkpasswd –l [-d] > /etc/passwd
mkpasswd –l [-d] > /etc/group
Note that the –d switch is necessary for domain users.

stephen_lin@abc-06d82fcbf1a /
$ ls –al
total 9
drwxr–xr-x  8 stephen_ mkpasswd    0 Jan 10 17:24 .
drwxr–xr-x  8 stephen_ mkpasswd    0 Jan 10 17:24 ..
drwxr–xr-x  2 stephen_ mkpasswd    0 Jan 10 19:48 bin
drwxr–xr-x  7 stephen_ mkpasswd    0 Jan 10 19:24 etc
-rw-r--r--  1 stephen_ mkpasswd  3262 Jan 10 22:03 insight.ico
drwxr–xr-x  5 stephen_ mkpasswd    0 Jan 10 19:48 lib
-rwxr–xr-x  1 stephen_ mkpasswd   53 Jan 10 22:03 moxa.bat
-rw-r--r--  1 stephen_ mkpasswd  3262 Jan 10 22:03 moxa.ico
drwxr–xr-x  2 stephen_ mkpasswd    0 Jan 10 19:48 tmp
drwxr–xr-x 14 stephen_ mkpasswd   0 Jan 10 19:48 usr
drwxr–xr-x  6 stephen_ mkpasswd    0 Jan 10 18:12 var

stephen_lin@abc-06d82fcbf1a /
$ _
```

Compilation

Windows tool chain is a cross compiler that can be used to compile Linux source code on a Windows operating system, allowing programmers to develop projects on a Windows-based PC. In this section, we describe the procedures you should follow to compile a project with the Windows tool chain.

First, copy the source codes of your project into the Windows tool chain's installation directory. In the example shown here, we want to compile the hello example provided in our CD-ROM. We will copy the source code to /UC which is the root directory of the Windows tool chain installation. Check in the root directory, to find the hello directory.

```
/
Mkpasswd –l [-d] > /etc/passwd
Mkgroup –l [-d] > /etc/group
Note that the –d switch is necessary for domain users.

Jared_Wu@Jared_Wu ~
$ cd /

Jared_Wu@Jared_Wu /
$ ls –al
total 9
drwxrwx---  9 Jared_Wu 4294967295    0 Nov  4 18:11 .
drwxrwx---  9 Jared_Wu 4294967295    0 Nov  4 18:11 .
drwxrwx---  2 Jared_Wu 4294967295    0 Nov  4 11:10 bin
drwxrwx---  7 Jared_Wu 4294967295    0 Nov  4 11:10 etc
drwx------  2 Jared_Wu mkpasswd    0 Nov  4 18:07 hello
-rwxr-x---  1 Jared_Wu 4294967295  3262 Nov  4 11:12 insight.ico
drwxrwx---  5 Jared_Wu 4294967295    0 Nov  4 11:10 lib
-rwxr-x---  1 Jared_Wu 4294967295   53 Nov  4 10:12 moxa.bat
-rwxr-x---  1 Jared_Wu 4294967295  3262 Nov  4 11:12 moxa.ico
drwxrwx---  4 Jared_Wu 4294967295    0 Nov  4 18:11 tmp
drwxrwx--- 13 Jared_Wu 4294967295    0 Nov  4 11:06 usr
drwxrwx---  6 Jared_Wu 4294967295    0 Nov  4 11:06 var

Jared_Wu@Jared_Wu /
$
```

Next, enter the hello directory and invoke the make program that will execute the compilation instructions written in the Makefile to compile the hello project.

```
/hello
drwxrwx---  7 Jared_Wu 4294967295    0 Nov  4 11:10 etc
drwx------  2 Jared_Wu mkpasswd    0 Nov  4 18:07 hello
-rwxr-x---  1 Jared_Wu 4294967295  3262 Nov  4 11:12 insight.ico
drwxrwx---  5 Jared_Wu 4294967295    0 Nov  4 11:10 lib
-rwxr-x---  1 Jared_Wu 4294967295   53 Nov  4 10:12 moxa.bat
-rwxr-x---  1 Jared_Wu 4294967295  3262 Nov  4 11:12 moxa.ico
drwxrwx---  4 Jared_Wu 4294967295    0 Nov  4 18:11 tmp
drwxrwx--- 13 Jared_Wu 4294967295    0 Nov  4 11:06 usr
drwxrwx---  6 Jared_Wu 4294967295    0 Nov  4 11:06 var

Jared_Wu@Jared_Wu /
$ cd hello/

Jared_Wu@Jared_Wu /hello
$ make
/usr/local/mxscaleb/bin/mxscaleb-gcc –o hello-release hello.c
/usr/local/mxscaleb/bin/mxscaleb-strip –s hello-release
/usr/local/mxscaleb/bin/mxscaleb-gcc –ggdb -o hello-debug hello.c

Jared_Wu@Jared_Wu /hello
$ ls
Makefile  README  hello-debug  hello-release  hello.c

Jared_Wu@Jared_Wu /hello
$
```

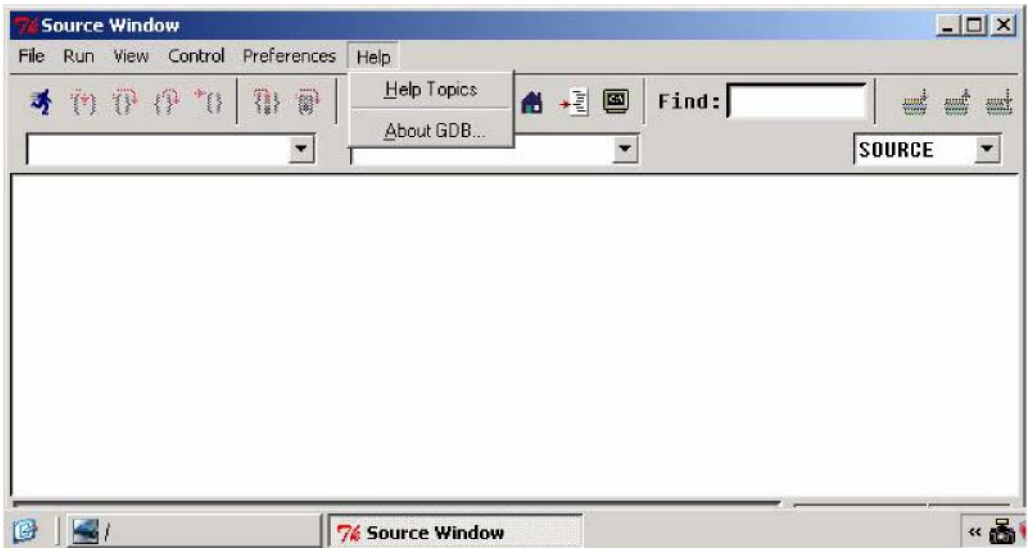
After that, use FTP to upload the executable file to the UC, and then run the executable.

```
/hello
ftp> bye

Jared_Wu@Jared_Wu /hello
$ ftp 192.168.14.9
Connected to 192.168.14.9.
220 Moxa FTP server <Version wu-2.6.1<2> Mon Nov 24 12:17:04 CST 2003> ready.
User <192.168.14.9:<none>>: root
331 Password required for root.
Password:
230 User root logged in.
ftp> bin
200 PORT command successful.
150 Opening BINARY mode data connection for hello-release.
226 Transfer complete.
ftp: 2744 bytes sent in 0.00Seconds 2744000.00Kbytes/sec.
ftp> bye
221-You have transferred 2744 bytes in 1 files.
221-Total traffic for this session was 3131 bytes in 1 transfers.
221-Thank you for using the FTP service on Moxa.
221 Goodbye.

Jared_Wu@Jared_Wu /hello
$

root@Moxa:~# chmod 777 hello-release
root@Moxa:~# ./hello-release
Hello
```

GDB Debug Tool—Insight

Insight is a graphical user interface that accompanies **GDB**, the GNU Debugger was written in Tcl/Tk by people working at Red Hat, Inc., and Cygnus Solutions. Red Hat was generous enough to make Insight available for public use, and continues to maintain the program.

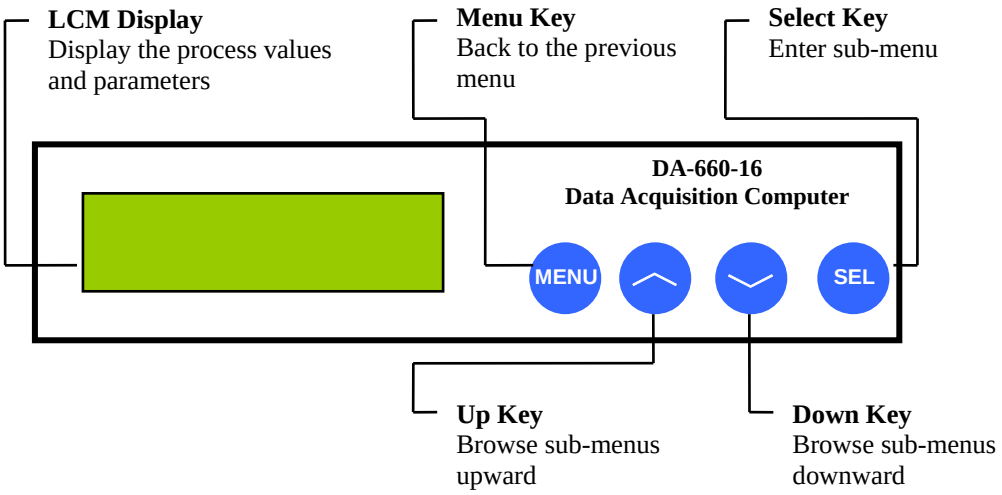
Click on http://sources.redhat.com/insight/ for more information about using Insight, or click on **Help Topics** under the **Help** menu to read the user manual.



C

Using the LCM Keypad

The Da-660 has an LCM and keypad that provide users with on-site parameter readings, and the ability to modify the configuration. Parameters that can be modified include model name, firmware version, network settings, etc. We use the DA-660-16-LX as an example to demonstrate the steps you should follow to configure these parameters.



Typical example:

- **Model Name and Firmware Version**—This is the 1st screen that is displayed when the system boots up.

  DA-660-16-LX
  Ver 1.0

  Press **MENU** to enter the main menu.

- **Main Menu**

  Network ↑
  Serial Port ↓

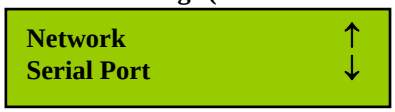
  Press ﹀ or use **SEL** to select an item.

  Serial Port ↑
  Console Port ↓

  Press ﹀ or use **SEL** to select an item.

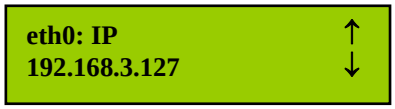
Press ^ or use **SEL** to select an item.

- **Network Settings (Port eth0 for example)**



Press **SEL.**

Press **SEL.**



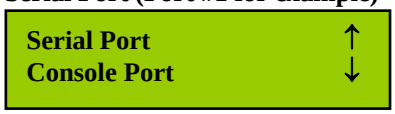
Press ˇ.

Press ˇ.

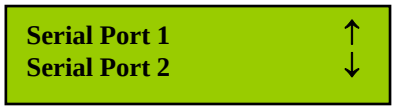
Press ˇ.

Press ^.

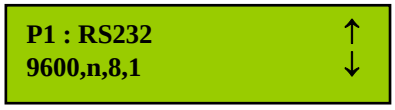
- **Serial Port (Port #1 for example)**



Press **SEL.**



Press **SEL.**



Press ˇ for port 2.

- **Console Port**

Press **SEL.**